Universidad Carlos III de Madrid

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN: SONIDO E IMAGEN

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

PROYECTO FIN DE CARRERA

Análisis de características relevantes en la emotividad de las canciones mediante aprendizaje máquina

Autor: **Alberto Sánchez Rojas**

Director: **Emilio Parrado Hernández**

Tutor: **José Miguel Leiva Murillo**

OCTUBRE, 2015

Agradecimientos

En primer lugar, me gustaría agradecer a Emilio la oportunidad que me ofreció para realizar este proyecto. Sus conocimientos y consejos han sido imprescindibles durante todo este tiempo.

Gracias a mi compañero y amigo Rubén, por ofrecerme su apoyo, ayuda y experiencia desde el primer día de trabajo, hasta los últimos momentos previos a la entrega.

A mi compañero de titulación Tomás por ayudarme en la parte más complicada, prestándome gran parte de su tiempo.

Gracias a mi familia, por estar siempre conmigo y cuidar de mi bienestar.

A mis amigos, muy especialmente a Óscar, Laura y Marcos, que siempre han hecho que me sienta capaz de conseguirlo y han creído en mis posibilidades.

A la gente del CAU, por aguantarme y darme ánimos todos los días durante este largo proceso.

Gracias a Irene, por compartirlo todo conmigo. Por ser la persona que siempre ha estado, y estará, en los buenos y sobre todo, en los malos momentos. Por ayudarme a superar todos los obstáculos que han surgido en estos meses, siempre con una sonrisa.

Y especialmente quería agradecer su comprensión y ayuda a José Miguel, por hacerse cargo de conducirme en los días más duros, previos a la defensa.

A todos ellos, y a los que me dejo sin mencionar, muchas gracias.

Resumen

El objetivo del presente proyecto fin de carrera es la identificación de las características musicales y sonoras que influyen en la emotividad alegre o triste de una canción. Para ello, se utilizan técnicas automáticas de extracción de características musicales, basadas en la teoría de Recuperación de Información Musical, e implementación de modelos lineales de clasificación.

El trabajo abarca las fases de diseño, programación en lenguaje MATLAB, simulación y experimentos para el aprendizaje supervisado de una Máquina de Soporte Vectorial.

Palabras clave: *características musicales, emotividad musical, extracción de características, Recuperación de Información Musical, modelos lineales, clasificación, decisión, aprendizaje supervisado, Máquinas de Soporte Vectorial*.

Abstract

The purpose of this final project is the identification of the musical and sound features that influences the happy or sad emotionality of a song. To do this, automatic techniques for music features extraction, based on the theory of Music Information Retrieval, and classification linear models implementation are used.

The work covers the design, MATLAB language programming, and simulation and experiments for a Support Vector Machine supervised learning phases.

Keywords: *musical features, musical emotionality, feature extraction, Music Information Retrieval, linear models, classification, decision, supervised learning, Support Vector Machines.*

Índice general

Índice de figuras

Capítulo 1. Introducción

1.1. Motivación

La música, en mayor o menor medida, forma parte de la vida diaria de las personas e interviene directamente en nuestras emociones y experiencias. La música está presente en casi todos los aspectos cotidianos: cine, televisión, internet, publicidad, entretenimiento, vida social, etc. La tendencia de los últimos años ha propiciado que cada vez consumamos más contenidos musicales y multimedia, en general. Los dispositivos que nos acompañan en nuestro día a día, como smartphones, tablets u ordenadores se han ido adaptando para poder almacenar, reproducir, editar y compartir todos estos contenidos musicales. Sin embargo, ¿qué información conocen estos dispositivos sobre esos archivos?

La manera más extendida por la cual las máquinas conocen las características de los archivos de música, es a través de metadatos o etiquetado de tipo textual. El problema de este tipo de métodos reside en que las descripciones las realizan las propias personas. Pongamos el ejemplo de la red social Last.fm, en la cual la gente escucha música en sus dispositivos, mientras publica comentarios sobre las canciones, artistas, fechas de conciertos, e incluso etiquetan, mediante los llamados *tags*, la música que escuchan. Esto resulta de gran ayuda para establecer una buena base de datos de canciones y artistas en torno a géneros musicales, por ejemplo. Sin embargo, el hecho de que sean las propias personas quienes etiqueten la música puede dar lugar a problemas, ya sea de manera malintencionada o por error, provocando un etiquetado erróneo de las canciones. Este hecho puede llevarnos a la siguiente pregunta: ¿y si desarrolláramos métodos automáticos para que los propios dispositivos sean capaces de distinguir el contenido de las canciones sin necesidad de información proporcionada por nosotros?

La motivación de este trabajo surge como respuesta a la pregunta anterior, aplicada al terreno de la emotividad en canciones. Si se descubre qué propiedades y características, intrínsecos a la música, son los que inducen las emociones en nuestro cerebro, se podría alcanzar cierto acercamiento de las emociones humanas a las máquinas. Este hecho podría tener, en un futuro a medio-largo plazo, numerosas aplicaciones en terrenos tan dispares como la inteligencia artificial, composición musical, musicoterapia, etc.

A día de hoy, la limitación de crear un escenario en el que las máquinas sean capaces de asimilar, en tiempo real, las emociones provocadas por la música, viene determinada por una combinación entre la complejidad de los algoritmos y la capacidad de proceso de los dispositivos de uso común. Esto puede que no sea una restricción en un plazo de tiempo considerable y, por lo tanto, se podría llegar a conseguir que las máquinas sean capaces de "sentir" la música.

1.2. Objetivos

El objetivo final de este proyecto es identificar qué características son más relevantes para determinar la emotividad de una canción. Otros objetivos que se deberán cumplir para lograrlo son:

- Comprender el funcionamiento de un modelo de clasificación mediante máquinas de vector soporte.
- Elegir implementaciones adecuadas para las técnicas de extracción y selección de características, así como para el entrenamiento y test de la SVM.
- Diseñar un código eficiente en MATLAB que reduzca el cómputo de las simulaciones lo máximo posible.
- Realizar los experimentos suficientes para garantizar resultados estadísticamente significantes.
- Estudiar la validez de los resultados con datos objetivos.
- Ofrecer líneas futuras del trabajo.

1.3. Contenido

La presente memoria se estructura en seis capítulos organizados de la siguiente manera:

- Capítulo 1. En él se ha expuesto una introducción al tema sobre el que se desarrolla el proyecto, detallando la motivación que ha llevado a su puesta en marcha y posterior realización, así como los objetivos que se pretenden alcanzar con la elaboración del trabajo.

- Capítulo 2. El segundo capítulo se dedica al estado del arte de los diferentes campos abarcados por este proyecto: tecnologías usadas, implementaciones de algoritmos y desarrollo de la teoría aplicada en los capítulos posteriores

- Capítulo 3. Este capítulo está basado en detallar cómo se ha organizado el desarrollo del proyecto, desde las primeras fases de documentación y planteamiento, hasta la programación del código del programa de simulación.

- Capítulo 4. El cuarto capítulo muestra los experimentos llevados a cabo tras el diseño del programa. En él se explican, en primer lugar, las condiciones y parámetros elegidos para cada uno de los experimentos. Posteriormente, se muestran los resultados obtenidos, se estudian las condiciones que han provocado dichos resultados, y se extraen las conclusiones pertinentes para cada caso.

- Capítulo 5. En este capítulo, se hace un resumen de todos los resultados arrojados durante la ejecución del proyecto, valorando si el objetivo del mismo ha sido alcanzado. Se expondrán los puntos clave que han llevado a estos resultados, así como los cambios que se podrían haber realizado para llegar a mejorar los presentes resultados.

- Capítulo 6. El último capítulo está dedicado a líneas futuras sobre el trabajo realizado. Se explicarán posibles modificaciones, ampliaciones o mejoras que puedan ayudar a continuar este trabajo de cara a mejorar los resultados obtenidos, o bien marcar nuevos objetivos a conseguir.

Se incluye, además, un anexo que muestra todas las canciones utilizadas para crear los conjuntos de datos empleados en la fase de experimentos.

Capítulo 2. Estado del arte

2.1. Recuperación de información musical

La Recuperación de Información Musical o, más ampliamente conocida como MIR, siglas de Music Information Retrieval, es un campo multidisciplinar que une técnicas como el procesado digital de señales de audio, reconocimiento de patrones, diseño de sistemas de software y aprendizaje máquina. Los algoritmos de recuperación de información musical permiten a los ordenadores, de alguna manera, "escuchar" o incluso comprender los datos contenidos en una pieza de audio, como, por ejemplo, archivos mp3 almacenados en una colección de música, audio en streaming o efectos de sonido.

De la misma manera que los humanos podemos reconocer características intrínsecas de un sonido o de la música, como el tempo, las notas musicales, acordes, géneros musicales o la estructura de una canción, los algoritmos MIR son capaces también de reconocer, extraer y almacenar esta información, permitiendo un procesado posterior y usos prácticamente ilimitados, tales como selección, búsqueda, recomendación, generación de metadatos, transcripción u optimización de sistemas en tiempo real [1].

Cualquier sonido o pieza musical contiene datos únicos que lo describen. Al trasladar esta información a un ordenador mediante algoritmos MIR, gracias a estos datos, se pueden analizar ciertas características únicas del sonido. Se puede pensar que esta información es como el "código genético" de un sonido. Todo sonido, por tanto, puede ser analizado basándose en estas características.

Podemos dar acceso a esta información a las máquinas, para que, de alguna manera, puedan entender las características del sonido, y enseñarles algoritmos para escuchar y distinguir sonidos, de una manera similar a cómo se comporta el oído humano, donde este proceso se realiza de manera natural e inconsciente. A su vez, las máquinas pueden utilizar algoritmos MIR para aprender las reglas de la música ofreciendo, por ejemplo, un gran abanico de posibilidades en el contexto de la composición musical [2].

Un desafío clave en el área de la información musical es el desarrollo de herramientas eficientes y fiables para la descripción, búsqueda y recuperación. Las técnicas convencionales utilizadas para llevar a cabo estas tareas están basadas, en su mayoría, en metadatos textuales. Sin embargo, el contenido musical y el contenido textual tienen una naturaleza muy diferente, que a menudo provoca que la recuperación de información mediante texto resulte insatisfactoria. Un objetivo principal de este campo, por lo tanto, es estudiar y desarrollar componentes innovadores para la descripción musical y los sistemas de recuperación de información, utilizando descriptores semánticos de contenido musical extraídos automáticamente mediante técnicas de reconocimiento de patrones y aprendizaje máquina [3].

La recuperación de información musical es un campo creciente, que cuenta en la actualidad con numerosas comunidades de investigadores en todo el mundo. Dos ejemplos son MIREX, Music Information Retrieval Evaluation eXchange, e ISMIR, la International Society for Music Information Retrieval, una organización no lucrativa que supervisa la organización de la Conferencia ISMIR. Esta conferencia se celebra anualmente y se trata de uno de los foros de investigación más importantes del mundo en procesamiento, búsqueda, organización y acceso a datos relacionados con la música [4] [5].

En la actualidad hay una gran cantidad de aplicaciones que hacen uso de alguna de las facetas de la recuperación de información musical. Dada su enorme utilidad, muchas de ellas se han convertido en herramientas imprescindibles para millones de usuarios y plataformas musicales en todo el mundo. A continuación se nombran algunas de las más destacadas, ordenadas por el tipo de técnicas que utilizan:

- Identificación de audio:
  Shazam
  Gracenote

- 'Score following':
  Rock Prodigy
  Smart Music
  Rockband

- Transcripción automática de música:
  Zenph

- Recomendaciones, 'Playlisting':
  YouTube Music
  Last.fm
  Pandora
  Spotify

2.1.1. MIRToolbox

Se ha expuesto anteriormente la motivación de los algoritmos de recuperación de información musical, así como sus usos y tendencia actual. En este apartado se expone una de las implementaciones de estas técnicas con un uso más extendido en investigación y desarrollo de aplicaciones de recuperación de información. Además, esta implementación ha sido la elegida para conformar parte del escenario de diseño, selección y extracción de características.

MIRToolbox es un conjunto integrado de funciones escrito en Matlab, dedicado a la extracción de características musicales de archivos de audio tales como el timbre, la tonalidad, el ritmo o la forma. Su objetivo es ofrecer una visión general, con un enfoque computacional, en el área de la recuperación de información musical [6].

El diseño es de tipo modular: los diferentes algoritmos se descomponen en fases, o bloques, formalizado mediante el uso de un conjunto básico de mecanismos elementales. Estos bloques forman el vocabulario básico de la toolbox, y pueden ser articulados de múltiples maneras, dejando libertad a los usuarios para que puedan seleccionar y parametrizar según sus criterios o necesidades. Además de los procesos computacionales básicos mencionados, se incluyen herramientas de extracción de características musicales de alto nivel, con estrategias y diferentes combinaciones que pueden ser seleccionadas por el usuario.

La elección de un diseño orientado a objetos permite una gran flexibilidad con respecto a la sintaxis: las herramientas se pueden combinar para formar conjuntos de métodos que corresponden a procesos básicos, como cálculo de espectro, autocorrelación, descomposición en ventanas, y características musicales. Estos métodos, además, pueden adaptarse a un amplio abanico de objetos de entrada a las funciones. Por ejemplo, el método de autocorrelación se comportará de una manera diferente con una señal de audio o una envolvente, pudiendo adaptar el resultado según el tipo de entrada.

MIRToolbox fue concebida inicialmente en el contexto del proyecto *Brain Tunning*, financiado por la Unión Europea (FP6-NEST). El principal objetivo del proyecto era investigar la relación entre las características musicales y las emociones inducidas por la música y la actividad neuronal asociada [7].

2.2. Clasificación. Modelos lineales.

En el presente trabajo se plantea un problema de clasificación, en el cual cada dato de entrada, en este caso una canción, debe ser asignado a una de las clases disponibles, representadas por una emotividad: triste o alegre.

El objetivo en clasificación es tomar un vector de entrada *x* y asignarlo a una de las *K* clases discretas C_k donde *k* = 1,. . ., *K.* El escenario más común presenta estas clases disjuntas, de manera que cada entrada se asigna a una sola de las clases. Por tanto, el espacio de entrada se divide en *regiones de decisión* cuyos límites se denominan *límites o superficies de decisión.* En un modelo lineal de clasificación se considera que las superficies de decisión son funciones lineales del vector de entrada *x* y se definen por *(D - 1)* hiperplanos, siendo el espacio de entrada de dimensión D. Los conjuntos de datos cuyas clases pueden ser perfectamente separadas por superficies de decisión se dice que son *linealmente separables* [8].

En clasificación, hay varias maneras de representar las etiquetas de las clases. Para modelos probabilísticos, la más conveniente, en el caso de problemas de dos clases, es la representación binaria en la que hay una sola variable $t \in \{0, 1\}$ que cumple *t=1* y representa la clase C_1 y $t = 0$ representa la clase C_2. Se puede interpretar el valor de t como la probabilidad de que la clase sea C_1, tomando únicamente los valores de probabilidad 0 y 1.

El enfoque más simple para problemas de clasificación consiste en diseñar una *función discriminante*, que asigna directamente cada vector *x* de entrada a una clase específica. Un enfoque más elaborado, sin embargo, se sirve de la distribución de probabilidad condicionada *p(C_k|x)* en una etapa de inferencia, que luego posteriormente se utiliza para tomar decisiones óptimas. Existen dos maneras de determinar la probabilidad condicionada *p(C_k|x)*. La primera se basaría en modelarla directamente, por ejemplo, con su representación mediante modelos paramétricos y la posterior optimización de los parámetros usando un conjunto de entrenamiento. Otra manera sería calculando las densidades dadas por *p(x|C_k)*, junto con las probabilidades a prori de las clases, *p(C_k),* para calcular las probabilidades a posteriori, utilizando el Teorema de Bayes:

$$p(C_k|x) = \frac{p(x|C_k)\, p(C_k)}{p(x)}$$

En los modelos de regresión lineal, la predicción de un modelo $y(x)$ viene dada por una combinación lineal de los parámetros *w*:

$$y(x) = w^T x + w_0$$

En problemas de clasificación, se pretende transformar $y(x)$ en un valor de probabilidad en el intervalo *(0,1)*. Para lograr esto, se considera una generalización del modelo en la cual se transforma la función lineal de *w*, utilizando una función no lineal $f(\cdot)$ de tal manera que

$$y(x) = f(w^T x + w_0)$$

En la literatura sobre aprendizaje máquina, $f(\cdot)$ se conoce como función de activación. Las superficies de decisión corresponden a $y(x) = constante$, por lo que $w^T x + w_0 = constante$ y por lo tanto las superficies de decisión son funciones lineales de *x*, incluso si la función $f(\cdot)$ es no lineal. La presencia de la función no lineal $f(\cdot)$ conduce a unas propiedades más complejas, analítica y computacionalmente.

2.2.1. Discriminantes lineales

Una función discriminante es una función que toma una vector de entrada, *x,* y lo asigna a una de las K clases C_k. En concreto, se denominan *funciones discriminantes lineales* a aquellas en las que las superficies de decisión son hiperplanos. A continuación, se estudia el caso de *K = 2* clases.

La representación más simple de una función discriminante lineal se obtiene tomando una función lineal del vector de entrada de la forma

$$y(x) = w^T x + w_0$$

donde ***w*** corresponde al *vector de pesos*, y w_0 es el *sesgo*. El valor negativo del *sesgo* es, a veces, llamado *umbral*. Un vector ***x*** de entrada se asigna a la clase C_1 si $y(x) \geq 0$ y a la clase C_2 en cualquier otro caso. Por tanto, la decisión límite está definida por la relación $y(x) = 0$, lo cual corresponde a un plano *(D-1)* dimensional con una entrada *D* dimensional. Si consideramos dos puntos x_A y x_B, ambos pertenecientes a la superficie de decisión, y debido a que $y(x_A) = y(x_A) = 0$, se tiene que $w^T(x_A - x_B) = 0$ y entonces el vector ***w*** es ortogonal a todo vector que se encuentre en dicha superficie de decisión, por lo tanto ***w*** determina la orientación de la superficie de decisión.

De igual manera, si ***x*** es un punto en la superficie de decisión, entonces $y(x) = 0$ y la distancia normal desde el origen a la superficie de decisión viene dada por

$$\frac{w^T x}{||w||} = -\frac{w_0}{||w||}$$

Se puede comprobar que el parámetro w_0 determina la posición de la superficie de decisión. Estas propiedades se ilustran en la siguiente figura, para D=2:

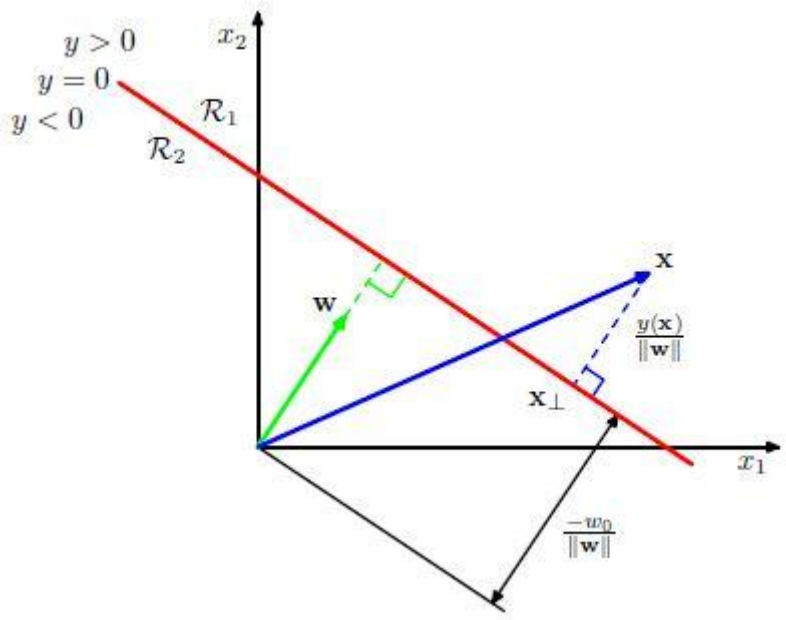

Figura 1: Ilustración de la geometría de una función discriminante lineal en dos dimensiones. La superficie de decisión, representada en rojo, es perpendicular a w, y su separación del origen está determinada por el parámetro w_0. Así mismo, la distancia perpendicular desde un punto x a la superficie de decisión está dada por $\frac{y(x)}{||w||}$

Se observa que el valor de $y(x)$ es una medida de la distancia perpendicular *r* del punto *x* a la superficie de decisión. Para ver esto, consideremos un punto arbitrario *x* tal que $x_\perp$es la proyección ortogonal sobre la superficie de decisión de tal manera que

$$x = x_\perp + r\frac{w}{||w||}$$

Multiplicando a ambos lados de esta ecuación por w^Ty sumando w_0, haciendo que $y(x) = w^T x + w_0$ y $y(x) = w^T x_\perp + w_0 = 0$, se tiene que

$$r = \frac{y(x)}{||w||}$$

resultado se puede ver ilustrado en la Figura 1.

En ocasiones es recomendable el uso de una notación más compacta, en la cual se introduce un valor de entrada $x_0 = 1$ y se define $\tilde{w} = (w_0, \boldsymbol{w})$ y $\tilde{x} = (x_0, \boldsymbol{x})$, obteniendo

$$y(x) = \tilde{w}^T \tilde{x}$$

En este caso, la superficie de decisión está formada por hiperplanos de dimensión D que pasan por el origen de un espacio de entrada de dimensión D+1.

2.2.2. Clasificación mediante máquinas de vector soporte

Una Máquina de Soporte Vectorial (SVM) aprende de la superficie de decisión de dos clases distintas de los puntos de entrada. Como un clasificador de una sola clase, la descripción dada por los datos de los vectores de soporte es capaz de formar una frontera de decisión alrededor del dominio de los datos de aprendizaje con muy poco o ningún conocimiento de los datos fuera de esta frontera. Los datos son mapeados por medio de un *kernel* Gaussiano o de otro tipo a un espacio de características en un espacio dimensional más alto, donde se busca la máxima separación entre clases. Esta función de frontera, cuando es traída de regreso al espacio de entrada, puede separar los datos en todas las clases distintas, cada una formando un agrupamiento [9].

La teoría de las máquinas de soporte vectorial (SVM por su nombre en inglés *Support Vector Machines*) son un conjunto de algoritmos de clasificación basados en la idea de minimización de riesgo estructural. En muchas aplicaciones, las SVM han mostrado tener gran desempeño, más que las máquinas de aprendizaje tradicional como las redes neuronales, y se utilizan como herramientas potentes para problemas de clasificación.

Una SVM primero mapea los puntos de entrada a un espacio de características de una dimensión mayor. Por ejemplo, si los puntos de entrada están en R^2, entonces son mapeados por la SVM a R^3 y encuentra un hiperplano que los separe y maximice el margen *m* entre las clases en este espacio, como se aprecia en la Figura 2.

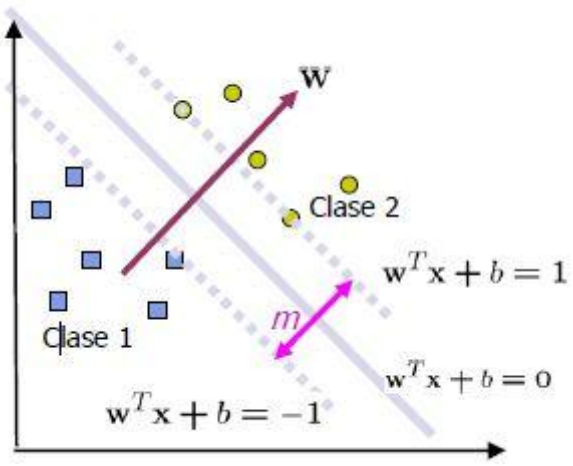

Figura 2: SVM: La frontera de decisión debe estar lo más lejos posible de los datos de entrada

Maximizar el margen *m* es un problema de programación cuadrática (QP) y puede ser resuelto por su problema dual introduciendo multiplicadores de Lagrange. Sin ningún conocimiento del mapeo, la SVM encuentra el hiperplano óptimo utilizando el producto punto con funciones en el espacio de características que son llamadas *kernels*. La solución del hiperplano óptimo puede ser escrita como la combinación de unos pocos puntos de entrada, llamados *vectores soporte*.

Caso linealmente separable

Se da un conjunto *S* de puntos etiquetados para entrenamiento, como se aprecia en la Figura 3

$$(y_1, x_1), \dots, (y_i, x_i) \qquad (1)$$

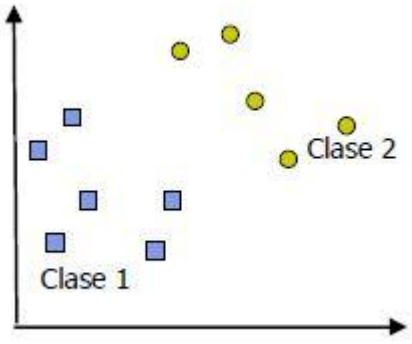

Figura 3: SVM: Caso linealmente separable

Cada punto de entrenamiento $x_i \in \mathbb{R}^N$ pertenece a alguna de las dos clases y se le ha dado una etiqueta $y_i \in \{-1,1\}$ para $i = 1, \dots, l$. En la mayoría de los casos, la búsqueda de un hiperplano adecuado en un espacio de entrada es demasiado restrictivo para ser de uso práctico. Una solución a esta situación es mapear el espacio de entrada en un espacio de características de una dimensión mayor y buscar el hiperplano óptimo allí. Sea $z = \varphi(X)$ la notación del correspondiente vector en el espacio de características con un mapeo φ de $\mathbb{R}^N$ a un espacio de características Z. Se desea encontrar el hiperplano

$$w\, z + b = 0 \qquad (2)$$

definido por el par (w, b), tal que se pueda separar el punto x_i de acuerdo a la función

$$f(x_i) = sign(w\, z_i + b) = \begin{cases} 1 & y_i = 1 \\ -1 & y_i = -1 \end{cases} \qquad (3)$$

donde $w \in Z$ y $b \in \mathbb{R}$. De manera más precisa, el conjunto S se dice que es linealmente separable si existe (w, b) tal que las inecuaciones

$$\begin{cases} (w\, z_i + b) \geq 1, & y_i = 1 \\ (w\, z_i + b) \leq -1, & y_i = -1 \end{cases} \quad i = 1, \dots, l \qquad (4)$$

sean válidas para todos los elementos del conjunto S. Para el caso linealmente separable de S, se puede encontrar un único hiperplano óptimo, para el cual, el margen entre las proyecciones de los puntos de entrenamiento de dos diferentes clases es maximizado.

Caso no linealmente separable

Si el conjunto S no es linealmente separable, se debe permitir infringir la formulación de la SVM

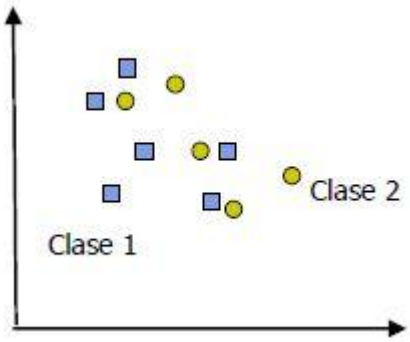

Figura 4: SVM: Caso no linealmente separable

Para tratar con datos que no son linealmente separables, el análisis previo puede ser generalizado introduciendo algunas variables no negativas $\xi_i \geq 0$ de tal modo que (4) pasa a

$$y_i(w\, z_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, l. \qquad (5)$$

Los $\xi_i \neq 0$ en (5) son aquellos para los cuales el punto x_i no satisface (4). Entonces el término $\sum_{i=1}^{l} \xi_i$ puede ser tomado como algún tipo de medida del error en la clasificación.

El problema del hiperplano óptimo es entonces redefinido como la solución al problema

$$min\left\{\frac{1}{2} w\, w + C \sum\nolimits_{i=1}^{l} \xi_i\right\}$$

$$sujeto\ a \quad y_i(w\, z_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, l \qquad (6)$$

$$\xi_i \geq 0, \quad i = 1, \dots, l$$

donde C es una constante. El parámetro C puede ser definido como un parámetro de regularización. Este es el único parámetro libre de ser ajustado en la formulación de la SVM. El ajuste de este parámetro puede hacer un balance entre la maximización del margen y las infracciones a dicho margen.

Buscar el hiperplano óptimo en (6) es un problema QP, que puede ser resuelto construyendo un Lagrangiano y transformándolo en el dual

$$máx\ \{\ W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j z_i z_j$$

$$sujeto\ a\ \ \sum_{i=1}^{l} \alpha_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, l$$

donde $\alpha = (\alpha_1, \dots, \alpha_i)$ es un vector de multiplicadores de Lagrange positivos asociados con las constantes en (5).

El teorema de Kuhn-Tucker juega un papel importante en la teoría de las SVM. De acuerdo a este teorema, la solución $\bar{\alpha}_i$ del problema (7) satisface:

$$\bar{\alpha}_i \left(y_i \left(\bar{w}\, z_i + \bar{b}\right) - 1 + \bar{\xi}_i\right) = 0, \quad i = 1, \dots, l \qquad (8)$$

$$(C - \bar{\alpha}_i)\xi_i = 0, \quad i = 1, \dots, l \qquad (9)$$

De esta igualdad se deduce que los únicos valores $\bar{\alpha}_i \neq 0$ (9) son aquellos que para las constantes en (5) son satisfechas con el signo de igualdad. El punto x_i correspondiente con $\bar{\alpha}_i > 0$ es llamado *vector de soporte.* Pero hay dos tipos de vector soporte en un caso no separable. En el caso $0 < \bar{\alpha}_i < 0$, el correspondiente vector de soporte x_i satisface las igualdades $y_i\left(\bar{w}\, z_i + \bar{b}\right) = 1$ y $\xi_i = 0$. En el caso $\bar{\alpha}_i = C$, el correspondiente ξ_i es diferente de cero y el correspondiente vector de soporte x_i no satisface (4). Nos referimos a estos vectores de soporte como errores. El punto x_i correspondiente con $\bar{\alpha}_i = 0$ es clasificado correctamente y está claramente alejado del margen de decisión, como se ve en la Figura 5.

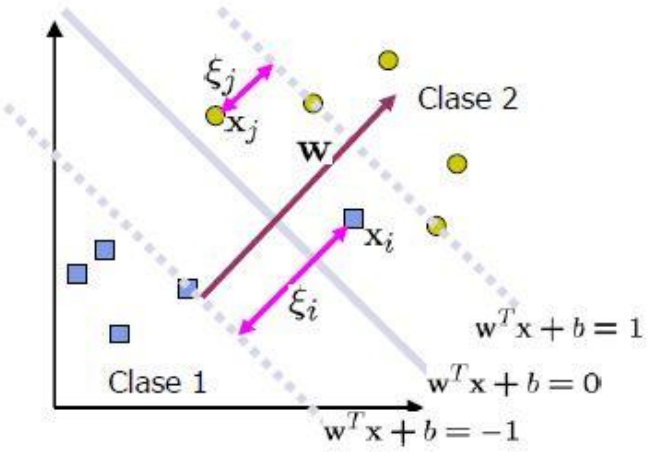

Figura 5: Aparición del parámetro de error ξ_i en el error de clasificación

Para construir el hiperplano óptimo $\overline{w}\ z + \overline{b}$, se utiliza

$$\overline{w} = \sum_{i=1}^{l} \overline{\alpha}_i\, y_i z_i \qquad (10)$$

y el escalar b puede ser determinado de las condiciones de Kuhn-Tucker (9).

La función de decisión generalizada de (3) y (10) es tal que

$$f(x) = sign(w\ z + b) = sign\left(\sum_{i=1}^{l} \alpha_i\, y_i z_i\ z + b\right) \qquad (11)$$

Capítulo 3. Diseño del proyecto

3.1. Planteamiento

Este capítulo se dedica a explicar detalladamente el proceso de planificación y creación del programa que servirá para llevar a cabo el objetivo del proyecto. En primer lugar, se mostrará, de manera resumida, las tareas que se pretenden cubrir, así como las herramientas de las que se disponen para su realización.

El primer paso consiste en obtener documentación suficiente sobre recuperación de información musical, y métodos de extracción y selección de características musicales. A su vez, se va conformando un conjunto de observaciones inicial consistente en catorce canciones en formato mp3.

Desde el principio se planteó el diseño del programa en Matlab, ya que es una herramienta que se ha utilizado ampliamente a lo largo de la carrera y se conoce en profundidad el lenguaje de programación. Se consideró adecuado su uso para trabajar con grandes cantidades de datos y la facilidad que ofrece para el tratamiento digital de la información.

Con estas premisas, la mayoría de búsquedas de información referidas a recuperación de información musical (MIR) y extracción de características dan con la herramienta MIRToolbox, un conjunto de funciones escritas en lenguaje Matlab, que facilita la tarea de extracción de características y automatiza los procesos de identificación y clasificación de las mismas.

Tras la familiarización con el software, se procede a una selección de características dentro del abanico ofrecido por MIRToolbox. Se observa que la cantidad de características a extraer es suficiente y que la integración de las funciones de extracción de cada característica con el resto de funciones a programar es cómoda.

Posteriormente, se automatiza un proceso para extraer todas las características de las canciones, y se crea, a partir de ellas, una matriz de datos con la que se puede pasar al siguiente paso.

Antes de pasar al aprendizaje de la máquina, se precisa de un conjunto de etiquetas, necesarias tanto para el entrenamiento, como para las pruebas. En este trabajo se ha simplificado el proceso de etiquetado, realizando un etiquetado subjetivo de todas las canciones. El proceso de etiquetado se describe de manera detallada en el siguiente capítulo, en los apartados *Creación del conjunto de datos*, ya que se ha adaptado a las necesidades de cada una de las pruebas realizadas.

El siguiente paso es la simulación de la máquina, proceso en el cual se ejecutan las siguientes tareas:

- Tratamiento de los datos (aleatorización, cálculo de parámetros y particionado)

- Entrenamiento
- Test
- Selección de características

Una de las implementaciones más extendidas de selección de características, y la que se utiliza durante el proceso, es la llamada *Recursive Feature Elimination*. Esta técnica sirve para reducir la dimensión del espacio de variables y eliminar variables irrelevantes o redundantes, mejorando la eficiencia del aprendizaje máquina. Consiste en evaluar el clasificador, eliminando recursivamente del conjunto cada una de las características, observando el error que se comete al quitar dicha característica.

Para las simulaciones se marca un número de iteraciones, con el objetivo de realizar un promediado posterior de los resultados de manera que sean estadísticamente significantes. Tras cada simulación, se evalúan tanto las prestaciones del clasificador, como la selección de características resultante, objetivo principal del proyecto.

Con un número suficientemente representativo de simulaciones, se procede a concluir cuáles son, en promedio, las características buscadas.

En el siguiente apartado, y en el Capítulo 4, se detalla cómo se ha llevado a cabo el proceso descrito.

3.2. Programación en Matlab

Para llevar a cabo este proyecto, se crean en Matlab dos programas diferenciados: uno se encarga de la extracción de las características y, a partir de sus valores numéricos, de la creación del conjunto de datos. El otro programa, al que se denominará 'programa principal', se encarga de leer las observaciones, preparar los datos y realizar las simulaciones pertinentes recurriendo al entrenamiento y test de la máquina, de manera iterativa.

3.2.1. Extracción de características y creación del conjunto de datos

Como se explicará en el próximo capítulo, la extracción se ha realizado con dos técnicas diferentes, según las necesidades y el escenario de observaciones:

- Para el primer conjunto de experimentos se recurre a una "extracción manual", llamando a través de un script a las funciones adecuadas de MIRToolbox para las características previamente seleccionadas.

- En el segundo conjunto de experimentos, las características se extraen mediante la función *mirfeatures,* disponible en MIRToolbox.

Posteriormente, se guardan los valores de las características de cada audio en vectores, en los cuales será de gran importancia la posición en la que ha sido almacenada cada una de ellas, ya que servirá después para conocer la relación número-característica de cara a la interpretación de resultados. Estos vectores, a su vez, se ordenan en una matriz que será uno de los parámetros de entrada en el programa principal.

3.2.2. Programa de simulación

Carga de datos e inicialización de variables.

Se cargan las variables *X* e *y* que son, respectivamente, la matriz que contiene todas las observaciones y el vector de etiquetas. También se inicializan las variables que contendrán los errores promedio y acumulado, el ranking de características y el número de iteraciones para cada simulación, valor que se establece como parámetro fundamental en las simulaciones, estudiado en el capítulo de experimentos, y que variará según las necesidades.

Aleatorización de datos.

Con el fin de conseguir mayor robustez en los experimentos, se aleatorizan los datos al principio de cada simulación. Para ello se realiza una permutación, generando un vector de números aleatorios sin repetición, del tamaño marcado por el número de filas de la matriz *X,* es decir, del número de observaciones. Este orden se aplica tanto a la matriz de datos, *X,* como al vector de etiquetas, *y*, de manera que sus filas quedan aleatorizadas con el mismo patrón.

Separación de conjuntos de entrenamiento y test.

En el siguiente paso se separan dos conjuntos a partir de todos los datos disponibles. Para el entrenamiento se toman el 80% de las observaciones, dejando el 20% restante para la fase de prueba o test.

Cálculo de parámetro C óptimo

C es el parámetro del clasificador que regula el compromiso entre sub-ajuste y sobre-ajuste en la frontera de decisión. Valores de C demasiado grandes o demasiado pequeños no se comportarán bien sobre los datos de validación: bien por sub-ajuste en el caso de C demasiado grandes, o por sobre-ajuste, en el caso de C pequeñas.

Para calcular el valor óptimo del parámetro dado un conjunto de datos, en primer lugar se toman los datos de entrenamiento y se divide en dos conjuntos a su vez, con un porcentaje establecido previamente. De esta manera, disponemos de un nuevo conjunto de entrenamiento y validación. Posteriormente, se normaliza el conjunto de datos y se establece un rango de valores para el parámetro C, el cual se irá validando en el siguiente paso.

Para encontrar el valor óptimo, se realiza un entrenamiento y predicción con los datos disponibles y se compara el error en cada uno de los casos. Finalmente, se elige el valor de C que menor error de predicción haya producido. Este valor se guarda para la simulación actual.

Normalización

Se normalizan los datos dejando cada variable con media nula y varianza unidad. Para ello, se calcula la media y la desviación típica de los datos de entrenamiento y se aplica la función *bsxfun*, consiguiendo un conjunto normalizado.

Algoritmo RFE

Una vez que se dispone del conjunto de datos separado y normalizado para la simulación actual, se comienza la selección de características con el algoritmo Recursive Feature Elimination. Este sistema de selección consiste en extraer, o eliminar, en cada iteración, la característica menos relevante en la decisión o, dicho de otra manera, la característica que provoca un error menor al eliminarla.

En cada iteración de RFE, se elimina la columna n-ésima de la matriz de datos de entrenamiento, es decir, los valores correspondientes a la característica número n, para evaluar el impacto que produce su ausencia en la clasificación. Tras ello, se pasa al proceso de entrenamiento y test, cálculo de error y descarte de característica, detallado a continuación:

Entrenamiento y test

Para la realización de los experimentos se han utilizado dos implementaciones diferentes para la simulación de la máquina de vector soporte, por razones que se explicarán más adelante. Como se verá en el siguiente capítulo, en el primer conjunto de datos se ha utilizado la implementación de aprendizaje máquina por defecto y, para el segundo conjunto, se opta por *libsvm* [10].

Conjunto 1: *svmtrain* y *svmclassify*

Tras extraer la característica asociada a la iteración, la máquina se entrena haciendo uso de la función *svmtrain* de Matlab, utilizando como entrada los datos restantes, el vector de etiquetas y el valor del parámetro C calculado previamente.

El siguiente paso consiste en validar la clasificación. Se toma el conjunto elegido para realizar el test y se eliminan los valores correspondientes a la característica que se está evaluando. Se llama a la función *svmclassify* con el modelo entrenado en el paso previo y los datos de test. Como resultado, se obtiene un vector que contiene la predicción de la máquina para los nuevos datos.

Conjunto 2: libsvm

Para crear el modelo, con libsvm se establecen una serie de opciones previas, que son los parámetros que utilizará la máquina para realizar el entrenamiento. En este caso, se utilizan los siguientes:

```
options = ['-t 0 -q',' -c ', num2str(C)];
```

donde

-t 0 establece que la función kernel sea lineal

-q fuerza a que no haya salidas por pantalla (quiet mode)

-c indica el valor del parámetro C calculado

En el siguiente paso, se entrena el modelo con el conjunto X de entrenamiento, las etiquetas de entrenamiento y las opciones anteriores, utilizando la función *svmtrain* de libsvm

```
model = svmtrain( double(ytrain), Xtr, options);
```

Posteriormente, la salida de la función *svmpredict* es la predicción que realiza el clasificador utilizando el modelo anterior y los datos de prueba o test

```
[prdc, ~, ~] = svmpredict( double(ytest), Xte, model);
```

Cálculo de error de predicción

Para tener una medida objetiva de evaluación, se calcula el error cometido por el clasificador al eliminar la característica n-ésima como la diferencia entre la predicción del modelo entrenado y las etiquetas del conjunto de test.

Descarte de característica

En cada iteración del algoritmo se descarta la característica que da lugar a un menor error de predicción cuando esta es eliminada, es decir, que al eliminar dicha característica, se comete poco error y, por tanto, tiene menor relevancia en la decisión. Una vez elegida la característica,

se elimina del conjunto inicial y se procede a la siguiente iteración. Este proceso de eliminación, entrenamiento, test, cálculo de error y descarte se repite tantas veces como número de características existen en el conjunto de datos.

Ranking de variables y orden final

Tras cada simulación, se guardan los índices (originales) y el orden en que las variables han sido descartadas por el algoritmo para, posteriormente, poder comparar con las demás simulaciones y ordenar las características por importancia en la clasificación. Estos rankings de variables se guardan hasta el final de todas las simulaciones.

Por último, se obtienen las posiciones en las que ha quedado cada variable. Se hace la media por variables y se ordenan de mayor a menor valor, de tal manera que las variables que han ido siendo descartadas en las últimas posiciones obtienen mayor valor y las que fueron descartadas en las primeras iteraciones obtendrán, en media, un menor valor. De esta manera, se obtiene el orden final según importancia.

Capítulo 4: Experimentos

4.1. Primera aproximación: conjunto de datos 1.

4.1.1. Creación del conjunto de datos 1

Selección de canciones

Se parte de catorce canciones en formato mp3, las cuales son elegidas aleatoriamente de un conjunto de canciones, sin tener en cuenta duración o género musical, de tal manera que la mitad sean “canciones alegres” y, la otra mitad, “canciones tristes”. Se encuentran detalladas en el Anexo A.

Estos catorce archivos mp3 son convertidos a formato WAV, ya que se han observado mejores resultados para este formato durante la fase de pruebas de la herramienta, por ejemplo, en cuanto a tiempo de computación, como se aprecia en la siguiente Figura:

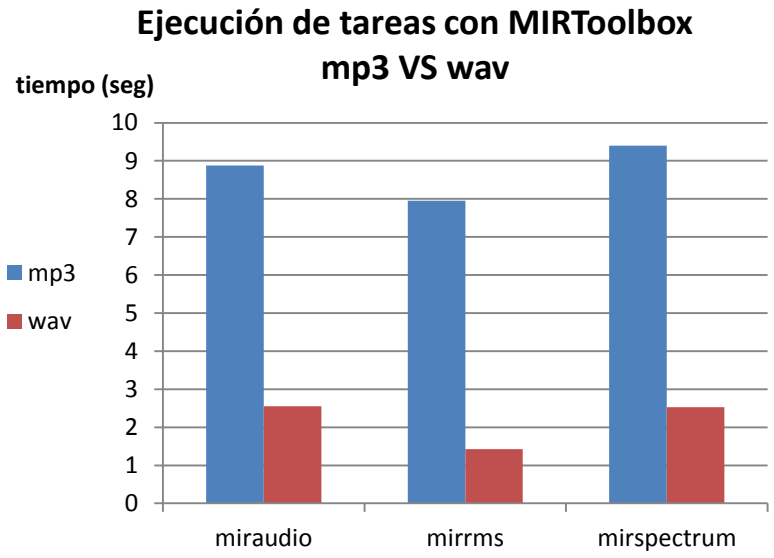

Figura 6: Comparativa de rendimiento mp3 VS WAV

Esto se debe a que el archivo en formato mp3, al ser un tipo de formato con compresión, necesita una descompresión previa al procesado, lo que incide directamente en el tiempo de ejecución de las tareas.

Selección y extracción de características

En la primera aproximación al problema, se utiliza la herramienta MIRToolbox para realizar la extracción de características. Esta herramienta permite, mediante diferentes funciones, obtener valores numéricos de ciertas características de una pieza de audio. Se elige esta herramienta por diversos motivos:

- Se recomienda su uso desde Mathworks para tareas de Music Information Retrieval.
- Dispone de características suficientes para la tarea que se pretende realizar.
- Se integra fácilmente en Matlab.
- Se observa un uso extendido de esta herramienta para este tipo de trabajos [11] [12] [13].

Una vez dispuestos los archivos de audio y la herramienta de extracción, se procede a elegir qué características de las ofrecidas en la toolbox se van a incorporar al proceso. Para ello, se realiza la extracción de cada una de las características mediante un pequeño fragmento de audio de prueba. Tras hacer un barrido a todas las características, se eligen aquellas que dan como resultado valores numéricos. La selección de características se muestra en la siguiente tabla:

Dinámica:	**Ritmo**	**Timbre**	**Tono**	**Tonalidad**	**Análisis**
rms	tempo	brightness	pitch	chromagram	entropy
lowenergy	metroid	rolloff	cepstrum	key	
envelope	fluctuation	mfcc		mode	
onsets	beatspectrum	inharmonicity		tonalcentroid	
attacktime	pulseclarity	roughness		hcdf	
attackslope		irregularity			
attackleap					
eventdensity					

Se descartan, por tanto, características como *metre*, que ofrece un resultado del tipo estructura de datos (struct) con una descripción detallada de la estructura métrica de un audio. Otra característica descartada es *midi*, que convierte un segmento de audio en una secuencia MIDI, lo cual no nos proporciona información relevante para la tarea que se desea desempeñar.

Por otra parte, se observa que determinadas características ofrecen una evolución temporal de la señal proporcionando múltiples valores, como por ejemplo *envelope* (envolvente de amplitud) u *onsets* (ráfagas de energía). Para tratar estas características "dinámicas", se estiman cuatro estadísticos del conjunto de valores para cada una de ellas. Por tanto, cada característica de este tipo da lugar a cuatro características diferentes: media, desviación típica, valor mínimo y valor máximo.

Se crea el programa *extractor.m* en Matlab, el cual pretende dar como resultado un vector de características a partir del nombre de un archivo de audio, de la siguiente manera:

x = extractor ('cancion.wav');

donde x es un vector fila de 66 elementos, los cuales corresponden a cada una de las características a extraer. Estos vectores darán lugar, posteriormente, a la matriz de observaciones. Para ello, se programa un script que llama a la función *extractor* tantas veces como canciones se han seleccionado.

Durante la ejecución de la extracción, se observa que Matlab se detiene por error de memoria. En las pruebas de extracción preliminares, el código había respondido correctamente en todas las ejecuciones. Sin embargo, las pruebas se realizaron con audios de test de duración más reducida, normalmente de veinte a treinta segundos. Se concluye que se alcanza el límite de uso de memoria permitido por el programa.

Para solventar este problema, se toma la decisión de trabajar con trozos de canciones, a los que se denominará segmentos, con una duración estimada como adecuada de veinte segundos. De esta manera, se toman las catorce canciones y se "trocean", dando lugar a 162 segmentos. Para la segmentación se utiliza la siguiente regla: Se van tomando trozos de veinte segundos del audio original y se van nombrando con el título de la canción seguido del número correspondiente. Al llegar al final de la canción, el último segmento se determina según los segundos sobrantes, de tal manera que si son menos de diez segundos, o diez inclusive, se añaden al segmento anterior y, si son más de diez, se crea uno nuevo aunque este no llegue a los veinte segundos. Se tienen por tanto, una mayoría de segmentos de veinte segundos y otros que varían de los once a los treinta segundos. Por ejemplo, una canción de duración 3:17 dará lugar a nueve segmentos de veinte segundos y uno de diecisiete; y otra de 3:46, diez segmentos y uno de veintiséis.

Conformado el nuevo conjunto de observaciones, se rehace el código del script para realizar la extracción de características de cada uno de los segmentos y posteriormente guardar los valores en una matriz, compuesta ahora de 162 filas y 66 columnas. En esta ocasión la ejecución no presenta errores y se obtiene la matriz de datos deseada.

Etiquetado de las observaciones

Dado el escenario de clasificación planteado, con dos clases, se procede a etiquetar las observaciones de cara al entrenamiento de la máquina. En principio, la decisión sobre el etiquetado era clara: etiqueta -1 para canciones tristes y etiqueta 1 para canciones alegres. De esta manera, quedaban perfectamente separados los dos conjuntos. Sin embargo, dada la necesidad de segmentar las canciones, descrita en el apartado anterior, este etiquetado deja de tener validez. Para ello, se introdujo la siguiente simplificación: Los segmentos provenientes de canciones tristes dan lugar a "segmentos tristes" y los de canciones alegres, a "segmentos alegres". De esta manera, se asigna la etiqueta -1 para segmentos tristes y la etiqueta 1 para segmentos alegres.

4.1.2. Simulaciones

En este punto, se dispone de una matriz de observaciones y un vector de etiquetas, por lo que se puede comenzar a realizar simulaciones. Se cargan estas variables al programa principal de simulación que se ha detallado en el capítulo 'Diseño del programa'. Como se ha observado durante las pruebas, el tiempo de simulación no es despreciable, invirtiendo de dos a cuatro horas en una simulación de cien iteraciones. Es un factor que parece decisivo en el desarrollo del proyecto, por lo tanto se empiezan las simulaciones con un número de iteraciones bajo. Se ejecuta el programa principal con el parámetro de iteraciones establecido a diez, en tres simulaciones independientes, para poder comparar los resultados de diferentes ejecuciones del programa. Tras la ejecución, los resultados que más interesan son el error cometido, o porcentaje de acierto del clasificador, y el orden de características, es decir, qué características han resultado ser más decisivas, en promedio, en el total de iteraciones.

Simulaciones 1, 2 y 3. Número de iteraciones: 10

La siguiente figura muestra la evolución del error cometido, en promedio, al ejecutar el programa en tres simulaciones independientes.

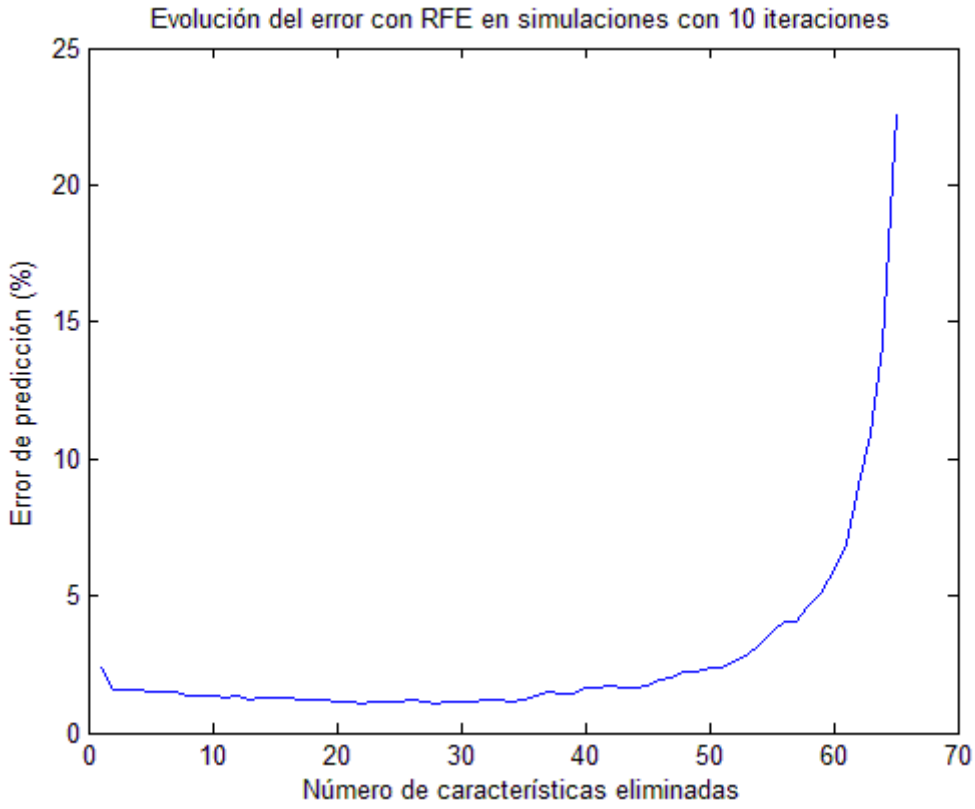

Figura 7: Evolución del error de clasificación en RFE promediado a 3 simulaciones de 10 iteraciones

Lo primero que se observa es el alto porcentaje de acierto en la clasificación, ya que en la mayoría de iteraciones el error está por debajo del 5%. La media del error resulta 2.90%.

Respecto a la evolución, se aprecia que el error va disminuyendo conforme se eliminan variables que se pueden llamar "ruidosas", hasta alcanzar un valor mínimo del error de 1.05% al haberse eliminado 28 características. Se puede apreciar cómo, a partir del momento en que se eliminan más de 50 características, el error se dispara debido a que el clasificador no tiene información suficiente para establecer una frontera de decisión. Pese a que, en las últimas iteraciones deberían quedar las "mejores" características, puede que estas se relacionen de alguna manera con otras que han sido eliminadas, provocando resultados no deseados.

Hasta aquí se han analizado los aspectos cuantitativos de error y la evolución del mismo durante el proceso de extracción de características. Una vez han sido realizadas varias simulaciones con los mismos parámetros, se pretende analizar el orden de importancia de características extraído de dichas simulaciones.

A modo de resumen, la siguiente tabla muestra el orden o ranking de características obtenido para cada una de las simulaciones. Por simplicidad, se muestran sólo las veinte primeras, ya que en las primeras posiciones se encuentran las que el clasificador ha determinado como más relevantes.

	Simulación 1	**Simulación 2**	**Simulación 3**
1	cepstrumMean	attackslopeMin	envelopeMin
2	fluctuation	hcdfMean	lowenergy
3	lowenergy	cepstrumMean	cepstrumMean
4	tonalcentroidMin	attackslopeMean	metroidStd
5	pulseclarity	chromagramMean	tempo
6	beatspectrumMin	tonalcentroidMin	metroidMin
7	mfccStd	envelopeMax	attackslopeStd
8	metroidMean	pitch	hcdfMean
9	mfccMean	beatspectrumMean	onsetsStd
10	onsetsMax	mode	Irregularity
11	attackslopeMin	tonalcentroidMax	rolloff
12	beatspectrumStd	chromagramMax	tonalcentroidMean
13	attackleapStd	onsetsStd	envelopeStd
14	attackleapMean	metroidMean	envelopeMean
15	chronogramMax	lowenergy	tonalcentroidMax
16	pitch	beatspectrumMin	beatspectrumMin
17	attackslopeMean	tempo	chromagramMax
18	irregularity	mfccStd	beatspectrumMax
19	envelopeMax	envelopeMin	attackleapMin
20	envelopeMin	mfccMean	beatspectrumStd

Se muestran los resultados de las simulaciones independientes para destacar el hecho de que, ante diferentes ejecuciones del programa, los rankings de relevancia resultan diferentes, aun habiendo entrenado con los mismos datos. Este hecho es un indicativo de que el valor elegido para el parámetro de número de iteraciones es demasiado bajo, ya que, con diez iteraciones no se consigue un orden de relevancia estable.

Se puede ver claramente en la tabla anterior que sólo la característica *cepstrumMean* parece destacar en las tres simulaciones. Esta característica corresponde a la variable *mircepstrum* de MIRToolbox, en concreto su valor medio, denominado *cepstrumMean* en el extractor de características. Esta característica trata de encontrar la frecuencia fundamental de la pieza de audio a través de una representación espectral.

La característica *lowenergy* también aparece en los primeros puestos en las simulaciones primera y tercera, aunque en la segunda simulación cae hasta el puesto quince. Esta corresponde a la variable *mirlowenergy* de MIRToolbox, denominada posteriormente *lowenergy* en el extractor de características. m*irlowenergy* ofrece una idea de la distribución temporal de la energía en la muestra de audio, mediante el porcentaje de 'frames' que permanecen por debajo del promedio de energía.

Tras las primeras simulaciones, se puede extraer que es altamente probable que la frecuencia fundamental y la distribución de energía del audio intervienen en la emotividad de una pieza musical. Sin embargo, los resultados no son demasiado consistentes ni se podría afirmar que son fiables, ya que el número de iteraciones es pequeño y el conjunto de datos es relativamente reducido. Se buscan entonces, simulaciones estadísticamente más significativas.

Para ello, en los siguientes experimentos, se decide aumentar el número de iteraciones en cada simulación, con el objetivo de 'afinar' los resultados y que adquieran mayor robustez. A continuación se muestran las simulaciones realizadas y los resultados para 100 iteraciones del algoritmo principal.

Simulaciones 4, 5 y 6. Número de iteraciones: 100.

A continuación se muestra la evolución del error, promediado con las tres simulaciones, utilizando cien iteraciones en cada una de ellas.

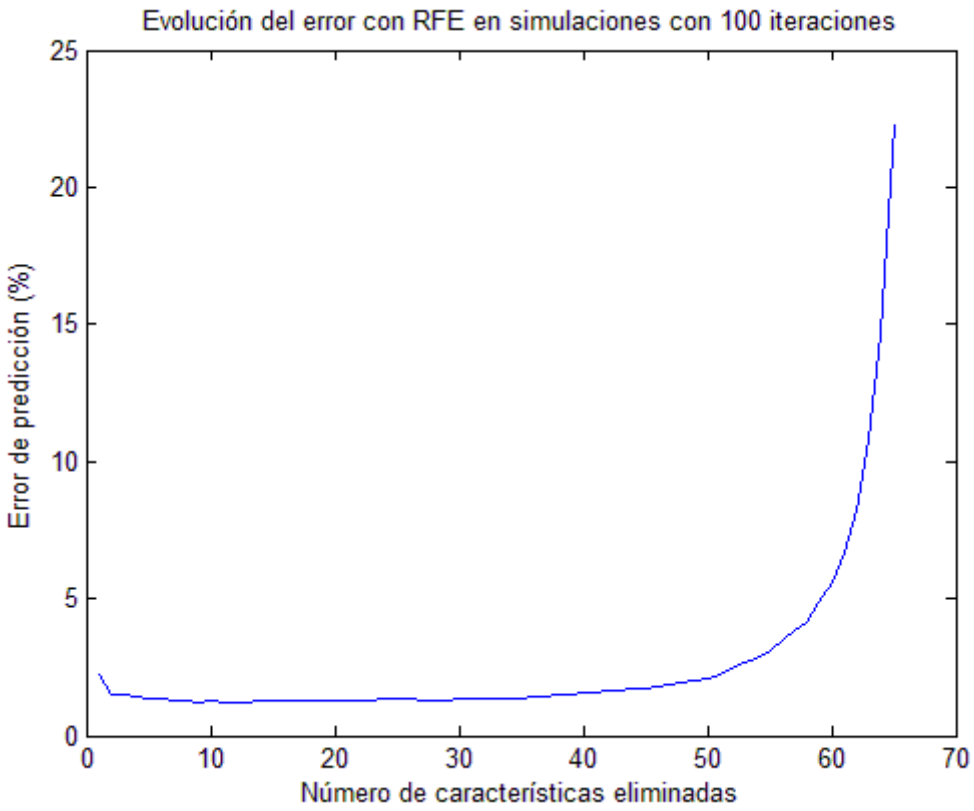

Figura 8: Evolución del error de clasificación en RFE promediado a 3 simulaciones de 100 iteraciones

Se observa una evolución muy similar al anterior conjunto de simulaciones. En este caso, el mínimo de error se alcanza al eliminar 13 características, llegando a un 1.23% de error. En media, el error resulta 2.83% un valor casi idéntico al obtenido en las primeras simulaciones.

En la siguiente figura se muestran las veinte características que ofrecen una mejor clasificación para este conjunto de canciones en las simulaciones con cien iteraciones:

	Simulación 4	**Simulación 5**	**Simulación 6**
1	cepstrumMean	cepstrumMean	cepstrumMean
2	pitch	pitch	pitch
3	envelopeMin	tonalcentroidMin	fluctuation
4	fluctuation	fluctuation	tonalcentroidMin
5	tonalcentroidMin	tempo	tonalcentroidStd
6	eventdensity	tonalcentroidStd	hcdfMin
7	irregularity	brightness	metroidMin
8	attackslopeMin	envelopeMin	attackslopeMean
9	tonalcentroidStd	pulseclarity	tempo
10	lowenergy	attackslopeMean	envelopeMin
11	mfccStd	beatspectrumStd	attackleapMean
12	attackslopeStd	lowenergy	attackslopeMin

13	attackslopeMean	tonalcentroidMean	onsetsStd
14	hcdfMin	roughnessMax	lowenergy
15	tempo	eventdensity	attacktimeMean
16	beatspectrumMean	hcdfMin	attackleapMin
17	mfccMax	mfccStd	chromagramMean
18	beatspectrumStd	metroidMin	roughnessMax
19	chromagramMean	attackslopeMin	onsetsMax
20	hcdfMax	attackslopeStd	tonalcentroidMean

Tras estas tres últimas simulaciones se aprecia claramente que, uno de los factores determinantes es el número de iteraciones del algoritmo, ya que en esta ocasión el ranking comienza a ser más estable en simulaciones independientes. Con un número suficientemente elevado, se puede converger a una solución aproximada al problema.

Como ya se vio en las primeras simulaciones, aquí se confirma que la característica *cepstrumMean* resulta la más importante para el clasificador, en este caso el promedio del valor numérico calculado mediante la extracción de la característica *mircepstrum*, o el valor promedio de la Transformada de Fourier del espectro de la señal. Como ya se vio en el anterior apartado, está íntimamente relacionada con la frecuencia fundamental del segmento musical.

En segundo lugar se encuentra la característica *pitch*, que corresponde a la característica *mirpitch* de MIRToolbox. En concreto, al pitch principal, ya que se ha establecido así en el extractor mediante las opciones de *mirpitch*, por simplicidad:

```
pitch = mirgetdata(mirpitch(audio,'Total',1)); % el pitch principal
```

Esta característica tiene mucho en común con la frecuencia fundamental de la señal de audio, aunque no es equivalente ya que en el pitch intervienen factores psicoacústicos subjetivos, debido a que se relaciona con la percepción que realiza el oído frente a un sonido.

Otra característica que se encuentra en los primeros puestos de importancia es *fluctuation*, la cual corresponde a *mirfluctuation* en MIRToolbox. Esta está definida por los autores de MIRToolbox como la "periodicidad rítmica a lo largo de los canales auditivos". Es una estimación del ritmo basada en un cálculo del espectrograma transformado por un modelado auditivo y una posterior estimación del espectro en bandas, en este caso de 10Hz.

La cuarta característica más destacada es *tonalcentroidMin*. Esta equivale a *mirtonalcentroid*, en concreto a su valor mínimo calculado en el extractor. Su valor viene determinado por un vector centroide tonal de seis dimensiones a partir del cronograma, el cual muestra una distribución de energía a lo largo de los diferentes pitches. Representa a una proyección de los acordes de quintas y terceras, mayores y menores.

A partir de la cuarta característica, hay diferencias notables en los rankings de variables. Otras características aparecen en los primeros puestos en alguna de las simulaciones. Sin embargo, esto no se repite en los tres experimentos, por lo que se han descartado como características relevantes.

Se ha comprobado que un aumento de iteraciones da lugar a un ranking más claro de características, pero a partir de este punto se plantea el problema del tiempo y la capacidad de cómputo. Para cada una de las simulaciones de cien iteraciones se requieren varias horas, por lo que se descarta aumentar este número por cuestiones prácticas. Se ha alcanzado un primer objetivo al identificar cuatro características que influyen decisivamente en la clasificación de la emotividad de piezas musicales. Por tanto, la primera aproximación se encuentra cubierta en este punto. En el próximo apartado de este capítulo se propondrán nuevos objetivos, aún más exigentes, para el segundo conjunto de datos.

Evaluación de resultados y conclusiones.

El primer factor a destacar en los experimentos es el número de iteraciones en cada simulación. Se ha podido comprobar cómo, aumentando este parámetro de diez a cien iteraciones, el clasificador es capaz de discriminar más claramente las características relevantes.

Con diez iteraciones por simulación se alcanza solamente una ligera idea de qué factores intervienen directamente en la emotividad de las canciones: parece que la frecuencia fundamental y la energía de la señal juegan un papel importante en la decisión, pero se echan en falta más datos concluyentes.

Al aumentar el número de iteraciones a cien se aprecia, sin lugar a dudas, que el clasificador es capaz de determinar con mayor precisión las características buscadas. No mejora en prestaciones, ya que el error cometido en las predicciones es mínimo en todos los experimentos. Sin embargo, sí aporta mayor precisión en la selección de características, ya que dispone de mayor número de datos con los que entrenar.

Al llegar a este punto, parece recomendable ir aumentando progresivamente el número de iteraciones y poder observar si se mantiene el ranking obtenido e incluso poder aumentar más allá de cuatro el número de características de las que se tienen certeza influyen en la clasificación. Pero se presenta la limitación del tiempo de ejecución, ya que con 200 o 500 iteraciones, el tiempo invertido en los experimentos sería demasiado elevado, y se busca modificar, en primer lugar, el conjunto de observaciones.

Otro aspecto a destacar es el error cometido por el clasificador, o dicho de otra manera, la tasa de acierto en las decisiones. Si se analiza detenidamente la matriz de errores en cada iteración del algoritmo se observan muchos errores que resultan igual a cero, de lo cual se pueden extraer varias conclusiones:

- Una de las variables que se utilizan para determinar qué característica se descarta en cada iteración es el mínimo de los errores cometidos al eliminar cada característica tras entrenamiento y test. Al existir múltiples errores igual a cero, se ha programado el algoritmo para que elija una de las características al azar dentro de todas las posibles, y de alguna manera, puede que esto influya en el resultado final, teniendo en cuenta que se cree que hay características que dependen unas de otras.

- Una de las lecturas que se puede extraer de este hecho es que, dado que existen muchos errores iguales a cero, significa que el problema planteado es separable. Es decir, que el clasificador casi siempre acierta y, por lo tanto, en condiciones de pocas iteraciones, resulta igual utilizar unas variables u otras.

Como se describió en el apartado 'Creación del conjunto de datos 1', se parte de un conjunto muy simplificado, basado en catorce canciones en las cuales resulta muy evidente la separación de la emotividad. Como se dijo, se eligieron canciones tristes y alegres a priori, para comprobar cómo se comportaba el clasificador.

Llegado este punto, y, dado que el aumento del número de iteraciones resulta incómodo y poco viable por las razones expuestas anteriormente, se toma la decisión de aumentar el conjunto de observaciones. El tiempo de ejecución se verá afectado también, sin embargo parece mejor estrategia modificar el conjunto de datos, aumentándolo en número y heterogeneidad, que continuar añadiendo iteraciones sobre el mismo conjunto.

Si es posible, se pretende aumentar el número de segmentos musicales por encima de mil, lo cual supone un importante salto cuantitativo. Además, también sería interesante introducir canciones donde la emotividad no sea claramente triste o alegre, para observar el comportamiento del clasificador ante datos más "inesperados" que en el primer conjunto.

Se desea, por tanto, la búsqueda de un modelo más real, que se aproxime lo más posible a un escenario en el que pueda introducirse cualquier canción, sin importar ningún parámetro preestablecido.

4.2. Segunda aproximación: conjunto de datos 2.

Como se ha concluido tras la primera aproximación al problema, la decisión tomada para continuar el trabajo ha sido crear un nuevo conjunto de datos con más observaciones. Se cree que un aumento en el número de observaciones puede dar lugar a un ranking de variables más fiable. Por el contrario, se estima que la tasa de acierto del clasificador disminuya, distanciándose de un escenario tan ideal como el visto en el primer apartado del capítulo.

4.2.1. Creación del conjunto de datos 2

Para esta fase de experimentos, se han investigado nuevos métodos, más eficaces, de creación del conjunto de datos, extracción de características e implementación de SVM para las simulaciones. Los cambios realizados, que se explicarán en los siguientes apartados, son:

- Aumentar el número de canciones hasta un total de 200, frente a las 14 utilizadas en la primera aproximación a la solución.
- Utilizar canciones completas como unidad de observación, en lugar de segmentos.
- Extracción de características mediante la función *mirfeatures*, disponible en MIRToolbox
- Entrenamiento y test con implementación *libsvm*, en lugar de *svmtrain* y *svmclassify*, funciones por defecto.

Selección de canciones

Durante la creación del conjunto de datos número 1 se explicó la razón de segmentar las canciones: el proceso de extracción realizado en Matlab presentaba errores por memoria insuficiente cuando se pretendían extraer las características de una canción completa.

En esta ocasión se ha decidido intentar utilizar canciones completas, en lugar de segmentos, ya que se tiene acceso a un equipo más potente y se cree que el inconveniente de procesamiento y memoria se verá solventado. De hecho, se piensa que trabajar con segmentos puede dar lugar a problemas a la hora de entrenar la máquina. El objetivo final del trabajo es descubrir qué características influyen más en la emotividad de una canción completa, no de partes de canciones. Cuando se entrena a la máquina con segmentos, está aprendiendo características que sólo se aplican en esos 20 segundos que duran los audios. La simplificación tomada en el primer conjunto respecto al etiquetado por segmentos puede dar lugar a una mala interpretación por parte de la máquina: si el segmento se está etiquetando, por ejemplo, como triste porque la canción es triste, la máquina puede estar aprendiendo de un dato erróneo, si resulta que el segmento tiene “características alegres”. Sin embargo, si se trabaja con canciones, etiquetando por tanto el audio completo, resulta más lógico asegurar que, en promedio, la canción es alegre o triste.

Se toman 200 canciones en formato mp3, elegidas al azar, de una biblioteca de música. Se incluyen géneros como rock, metal, pop, instrumental, clásica, acústica, etc. En esta ocasión no se han elegido expresamente canciones tristes y alegres, ya que se pretende que el escenario para la clasificación sea menos controlado y más real. La selección de canciones se puede consultar en el Anexo B.

El siguiente paso consiste en convertir los archivos a formato WAV, por la mayor eficacia observada (ver Figura 6). Dado que el conjunto es mucho mayor, se automatiza el proceso de conversión creando un script que utiliza la herramienta *ffmpeg* [14].

Selección y extracción de características

El extractor de características elegido para esta ocasión es la función *mirfeatures*, incluida en MIRToolbox. Esta función permite, de una manera optimizada, extraer características relevantes de un archivo de audio y sus estadísticos.

Las características que permite extraer la función *mirfeatures* se detallan en la siguiente tabla:

Dinámica	**Fluctuación**	**Ritmo**	**Espectro**	**Timbre**	**Tono**
rms	peak	tempo	centroid	zerocross	chromagram.peakPos
		attacktime	brightness	lowenergy	chromagram.peakMag
		attackslope	spread	spectralflux	chromagram.centroid
			skewness		keyclarity
			kurtosis		mode
			rolloff95		hcdf
			rolloff85		
			spectentropy		
			flatness		
			roughness		
			irregularity		
			mfcc		
			dmfcc		
			ddmfcc		

Para cada una de las 28 características, la función calcula seis estadísticos, de los que se eligen la media y la desviación típica como valores más representativos. Por tanto, a partir de cada característica se obtienen dos valores, que son considerados como características independientes. De esta manera, el número de características con las que se trabaja en esta ocasión es de 56.

Para proceder a la extracción, se crea un script el cual ejecuta la función *mirfeatures* tantas veces como canciones componen la base de datos, es decir, 200.

```
caracteristicas = mirfeatures( 'canción.wav', 'stat' );
```

De esta manera, se generan 200 variables de tipo *struct* que contienen la información que posteriormente será estudiada.

El siguiente paso consiste en ordenar los valores obtenidos en una matriz de datos. Para ello, se crea una función que lee los valores que se desean de cada característica, y los ordena de manera que las filas de matriz resultante son los audios y en las columnas se colocan las características, de la siguiente manera:

```
X(i,31) = songFeatures.spectral.irregularity.Mean;
X(i,32) = songFeatures.spectral.irregularity.Std;
```

Etiquetado de las observaciones

Como se ha comentado en la introducción del apartado, el proceso de etiquetado ha variado respecto a la primera aproximación. Se escuchan las canciones y se realiza un etiquetado subjetivo teniendo en cuenta el total de la canción, es decir, aplicando la emotividad que genera, en promedio, la canción completa. Se mantiene la asignación -1 para canciones tristes y 1 para canciones alegres.

4.2.2. Simulaciones

En este apartado también se introducen cambios respecto a los experimentos anteriores. Se intenta simular con los nuevos datos y la implementación de SVM utilizada hasta el momento y se observa que se presentan problemas de convergencia en *svmtrain* al intentar validar el parámetro C óptimo para la máquina.

Por este motivo, se busca una alternativa más eficiente a las funciones por defecto de MIRToolbox, encontrando la implementación *libsvm* como mejor opción. Tras las primeras pruebas, la validación del parámetro C se realiza sin problemas y el tiempo de simulación disminuye notablemente.

Simulaciones 1 a 5. Número de iteraciones: 100

Se ejecuta el programa principal un total de cinco veces. La siguiente figura muestra la evolución del error durante la ejecución del método RFE, promediado a las cinco simulaciones.

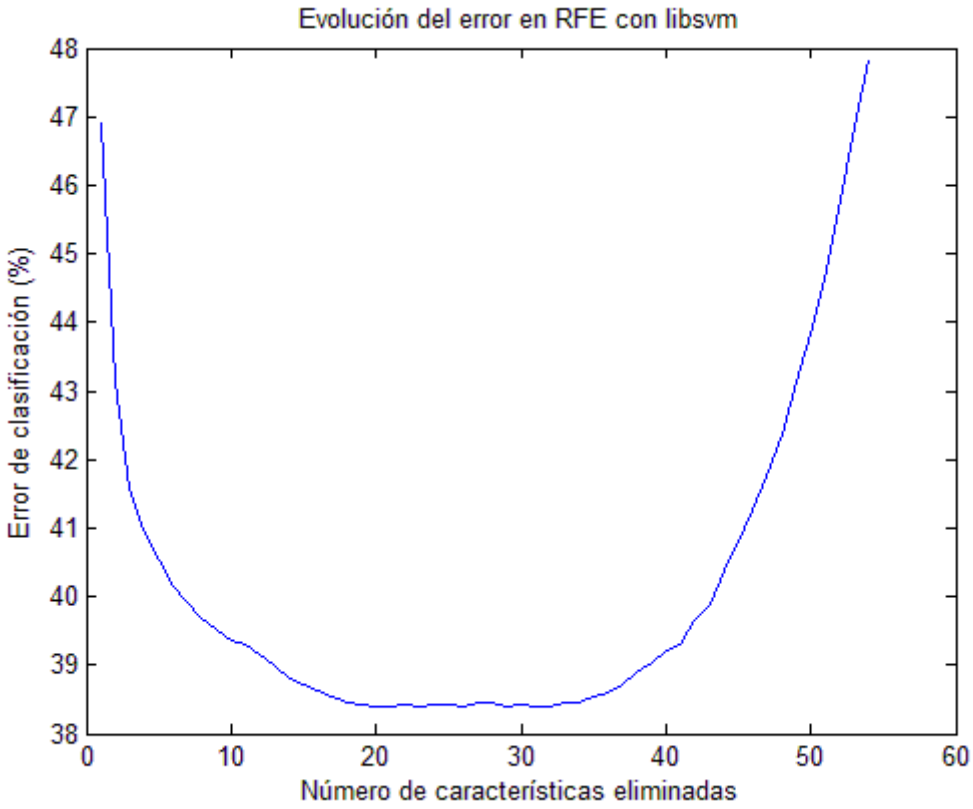

Figura 9: Evolución de error de clasificación en RFE con libsvm promediado a 5 simulaciones de 100 iteraciones

En primer lugar, y en comparación con el primer grupo de experimentos, llama la atención el aumento del error, encontrándose para todos los valores por encima del 38%. Ya se comentó anteriormente que era uno de los resultados esperados, debido al aumento de observaciones. El hecho de disponer de un escenario de observaciones más realista se ve reflejado en la media de error, que aumenta hasta el 40.26%.

Respecto a la evolución se puede decir que, en esta ocasión, la curva presenta una forma más lógica, ya que comienza con valores altos, cercanos al 50%, debido a que las características ruidosas provocan que el clasificador elija de una manera prácticamente aleatoria. Conforme se van eliminando las características ruidosas, el clasificador va construyendo una frontera de decisión más estable, hasta alcanzar el número óptimo de características, que, en este caso resulta ser 26, alcanzando un valor mínimo de error del 38.4%. Esto significa que se han eliminado 25 características que entorpecen la clasificación, y que se consigue la mejor clasificación utilizando 31 características de las seleccionadas. A partir de la eliminación de la siguiente característica, el error vuelve a aumentar ya que se descartan características de mayor o menor relevancia, pero, de alguna manera, importantes para el clasificador. Cuando se eliminan 40 características o más, el decisor tiende nuevamente a una situación de clasificación aleatoria, al no disponer de información suficiente.

A continuación, se muestra un resumen con la clasificación promediada de las características que han resultado más relevantes en las cinco simulaciones:

	Simulaciones 1-5
1	peak.PeakPosMean
2	dmfcc.Mean
3	irregularity.Std
4	keyclarity.Std
5	mfcc.Mean
6	spectralflux.Std
7	chromagram.peak.PeakMagMean
8	ddmfcc.Std
9	zerocross.Std
10	chromagram.peak.PeakPosMean
11	ddmfcc.Mean
12	irregularity.Mean
13	mode.Std
14	dmfcc.Std
15	tempo.Std
16	chromagram.peak.PeakMagStd
17	hcdf.Std
18	flatness.Std
19	chromagram.peak.PeakPosStd
20	brightness.Mean

Se han observado unos rankings muy parecidos en las cinco simulaciones. Si se compara este hecho con lo observado en las simulaciones de la primera aproximación, se puede deducir que este clasificador se comporta mejor, ya que las clasificaciones de características resultan muy similares en simulaciones independientes. Ante un mismo número de iteraciones del programa, esto se puede explicar con el cambio de estrategia en la elección de las observaciones y el aumento de número de las mismas, ya que la máquina tiene más datos para entrenar y más diferentes entre sí.

El ranking no varía apenas de una simulación a otra, por lo que se podría establecer un orden definitivo de importancia. Sin embargo, como en experimentos anteriores se ha demostrado que un aumento de iteraciones consigue unos rankings más estables, se procede a aumentar este parámetro en las siguientes simulaciones.

Simulación 6. Número de iteraciones: 500

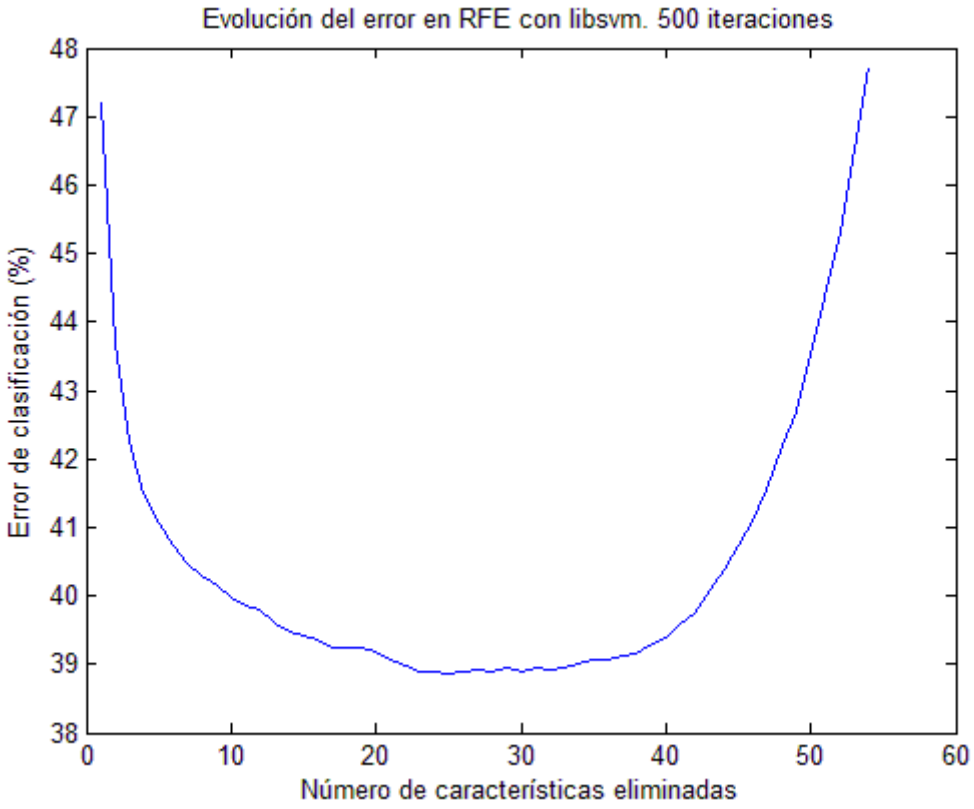

Figura 10: Evolución de error de clasificación en RFE con libsvm en simulación de 100 iteraciones

Para esta simulación se obtiene una evolución del error casi idéntica al caso anterior. El mínimo se sitúa en 38.86%, alcanzado al eliminar 25 características, y el valor medio en esta ocasión es de 40.61%.

El orden de características ha resultado ser el siguiente:

	Simulación 6
1	peak.PeakPosMean
2	irregularity.Std
3	dmfcc.Mean
4	keyclarity.Std
5	mfcc.Mean
6	spectralflux.Std
7	ddmfcc.Std
8	chromagram.peak.PeakMagMean
9	zerocross.Std
10	mode.Std
11	irregularity.Mean
12	chromagram.peak.PeakPosMean

13 dmfcc.Std
14 ddmfcc.Mean
15 brightness.Mean
16 hcdf.Std
17 chromagram.peak.PeakMagStd
18 flatness.Std
19 chromagram.peak.PeakPosStd
20 tempo.Std

Si se compara este ranking con el anterior, de simulaciones de 100 iteraciones, se aprecia que el orden apenas ha cambiado, pese a promediar esta vez con 500 resultados. En las nueve primeras posiciones aparecen las mismas características en ambas simulaciones. Parece que se está convergiendo a la solución final del problema. Por obtener un resultado aún más estadísticamente significativo, se ejecuta una última simulación que promedie 1.000 resultados del programa principal.

Simulación 7. Número de iteraciones: 1.000

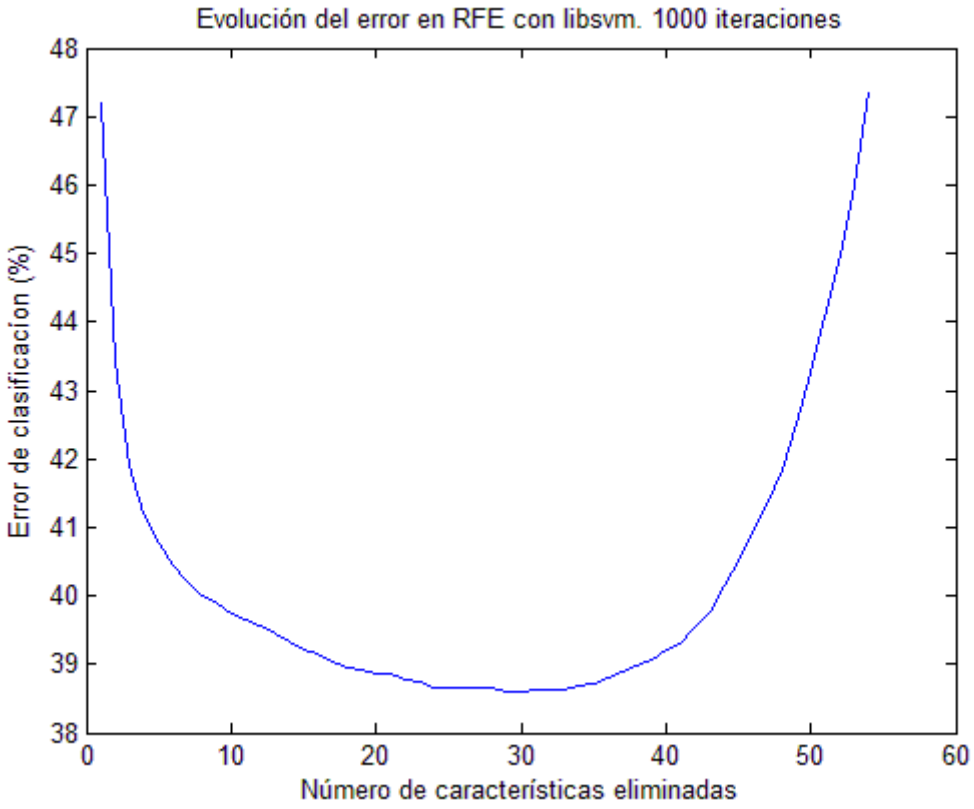

Figura 11: Evolución de error de clasificación en RFE con libsvm en simulación de 1000 iteraciones

Nuevamente, se obtiene una evolución muy parecida a las simulaciones anteriores. El valor mínimo se mantiene, resultando 38.60%, pero esta vez desplazando el número de características eliminadas para conseguirlo hasta 30. La media de error es de 40.36%.

El orden de características más relevantes tras 1.000 iteraciones se muestra en la siguiente tabla:

	Simulación 7
1	peak.PeakPosMean
2	irregularity.Std
3	dmfcc.Mean
4	keyclarity.Std
5	mfcc.Mean
6	spectralflux.Std
7	chromagram.peak.PeakMagMean
8	zerocross.Std
9	ddmfcc.Std
10	chromagram.peak.PeakPosMean
11	mode.Std
12	irregularity.Mean
13	ddmfcc.Mean
14	dmfcc.Std
15	hcdf.Std
16	tempo.Std
17	attack.slope.Mean
18	flatness.Std
19	brightness.Mean
20	chromagram.peak.PeakMagStd

Tras promediar con el doble de iteraciones el ranking de características se mantiene, resultando las seis primeras posiciones idénticas al experimento anterior.

Se considera, en este punto, que se han realizado simulaciones con promediados suficientes, sobre un espacio de observaciones amplio, como para extraer conclusiones fiables sobre la importancia de las características en la emotividad de una canción.

La característica más relevante, según las simulaciones, es ***peak***, en concreto *fluctuation.peak.PeakPosMean*. La fluctuación ya aparecía como característica importante en los primeros experimentos. Ahora se confirma, con un mayor número de datos como respaldo. Se añade, además, a esta característica el cálculo de la posición de los máximos locales mediante *mirpeaks*, en promedio. Parece, por tanto, que el clasificador necesita conocer la posición de los máximos durante la evolución del ritmo en la canción.

La segunda característica más importante resulta ser ***irregularity***, en concreto el valor de su desviación estándar. La irregularidad de un espectro es el grado de variación de los picos sucesivos del mismo. Por tanto, mide cuánto de variables son los máximos de la señal en su forma espectral.

La tercera posición por importancia corresponde al valor medio de la característica ***dmfcc***, que es la descomposición en ventanas de la variable *mfcc*, abreviación de mel-frequency cepstral coefficients. La descomposición se realiza, por defecto, en ventanas de 50ms con la mitad de superposición y con diferenciaciones temporales de orden d de los coeficientes. En el caso de *dmfcc*, d=1. Esta característica ofrece una descripción de la forma espectral del sonido, donde las bandas de frecuencia se colocan de manera logarítmica, en la escala de Mel, que se aproxima más a la respuesta del sistema auditivo humano que las bandas de frecuencia linealmente espaciadas.

La cuarta característica, por orden de relevancia, es la varianza de ***keyclarity***. Esta característica es una de las salidas posibles de la variable *mirkey*, y mide la fuerza (*mirkeystrength*) de la 'mejor' nota o nota predominante. *mirkeystrength* mide la probabilidad de las notas 'candidatas' o notas presentes en la pieza musical. De esta manera, el clasificador puede conocer la nota con más fuerza, en promedio, en la canción.

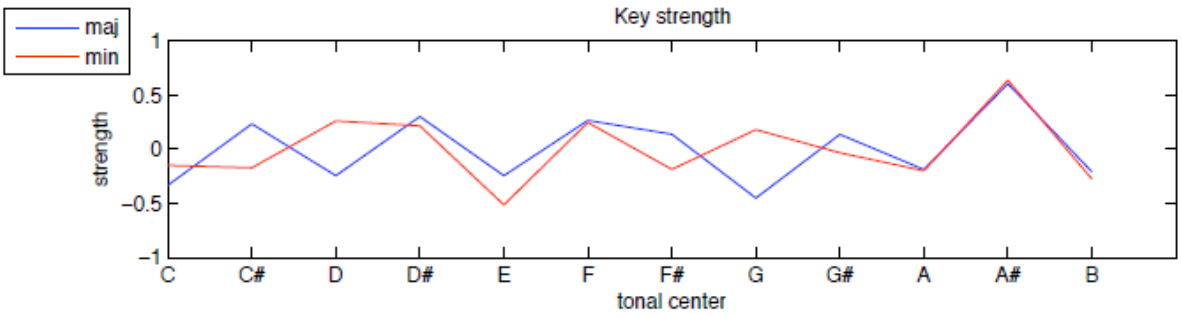

Figura 12: Representación de la función mirkeystrength

La quinta característica está relacionada con la tercera, pero en su versión sin descomposición en ventanas: el valor medio de mfcc. Como se ha visto antes, corresponde con una descripción de la forma espectral del sonido en bandas Mel.

La sexta característica más relevante es spectralflux, más concretamente su desviación estándar. Corresponde a la versión enventanada de mirflux. El flujo espectral se puede definir como la distancia entre eventos importantes en el espectrograma. Las fluctuaciones en los canales cuyas frecuencias están alrededor de 1600Hz a 6400Hz representan la percepción correspondiente a la "actividad" de la música.

Evaluación de resultados y conclusiones.

Tras este segundo conjunto de experimentos se ha conseguido alcanzar uno de los objetivos del proyecto: identificar qué características resultan más relevantes en un clasificador de emotividad en canciones.

Después de analizar los resultados con el segundo conjunto de datos, se cree que el cambio de estrategia en la elección de la unidad de observación, de segmentos a canciones enteras, ha sido acertado tomando como referencia el objetivo principal del trabajo.

Se ha podido comprobar cómo la implementación *libsvm* se comporta, con el mismo conjunto de datos, mejor que *svmtrain* y *svmclassify* en términos de convergencia, uso de memoria y velocidad de cómputo en general.

Respecto al conjunto de datos, resulta lógico asegurar que cuanto mayor número de observaciones, y más diversas se utilicen para entrenar la máquina, mejores resultados se obtienen. El problema que surge de un conjunto como el utilizado en la segunda aproximación, con el escenario de dos clases fijado por el problema, es el etiquetado. Resulta complicado etiquetar algunas canciones solamente en dos clases, ya que pueden evocar otras emotividades que no sean las expuestas en este proyecto. Esta es una de las causas por las que la tasa de acierto del clasificador haya disminuido hasta valores del 60% en promedio.

Las características seleccionadas por el clasificador otorgan una importancia mayor a la variación de la actividad en las canciones. Resultan más relevantes las características que ofrecen información sobre los máximos de la señal, los cambios en frecuencia y variaciones en el espectro. De alguna manera, son características que miden cuánto de cambiante es una canción.

Capítulo 5. Conclusiones

5.1. Conclusiones generales

En el presente capítulo, se procede a comentar las conclusiones generales derivadas de la realización del proyecto. Se sacarán conclusiones del planteamiento inicial aplicado, de los resultados intermedios y finales de los experimentos, así como del nivel de éxito alcanzado respecto a los objetivos marcados en el capítulo primero de esta memoria.

En primer lugar, respecto a las técnicas utilizadas, cabe mencionar el uso de la herramienta MIRToolbox, lo cual ha supuesto un gran acierto y una pieza fundamental en el desarrollo del trabajo. La facilidad con la que se obtienen valores para las características buscadas gracias a MIRToolbox, y su cómoda integración con el resto del programa, ha sido de gran ayuda en el desarrollo del proyecto. Sin esta herramienta, la extracción de las características de una manera "manual" hubiera sido una tarea extremadamente laboriosa y de una complejidad superior a la marcada por este trabajo. Por lo tanto, se puede concluir que MIRToolbox es esencial para un trabajo de estas características, basado en Recuperación de Información Musical.

Uno de los puntos clave del proyecto, y de los más delicados, es el etiquetado de las observaciones. Como se ha descrito en los capítulos 3 y 4, se tomó la decisión de realizar un etiquetado subjetivo sobre el conjunto de canciones. Este hecho tiene su parte positiva, pero plantea problemas teóricos y prácticos sobre su validez. Por un lado, la simplicidad de un etiquetado subjetivo facilita la labor de diseño. Por el contrario, un etiquetado basado en encuestas o a través de alguna aplicación de cualquier tipo hubiera conseguido unas etiquetas estadísticamente más representativas. En el capítulo siguiente se abordará este tema en favor de una posible ampliación de este trabajo mejorando el proceso de etiquetado.

El algoritmo de selección de características utilizado, Recursive Feature Elimination basado en SVM, ha resultado funcionar correctamente para el escenario disponible, provocando un mínimo impacto en la complejidad del código programado. Así pues, se recomienda el uso de esta técnica para labores de selección de características eficientes.

Respecto a la implementación de la extracción de características, se puede concluir que resulta más adecuada la opción de extracción automatizada que ofrece *mirfeatures*, detallada en el Capítulo 4, frente a la selección y extracción manual por "módulos". La primera alternativa muestra mayor compactación de los grupos de características y reduce el tiempo de ejecución de la extracción, ya que su salida son características que previamente se han seleccionado por los desarrolladores del programa. Por el contrario, si se desea un conjunto reducido de características, o bien el resultado de alguna operación no ofrecida en *mirfeatures*, la extracción por funciones o módulos, resulta igualmente eficaz, pero, como se ha dicho, menos indicada para extracciones de carácter general, como es el caso de este trabajo.

Otra conclusión interesante extraída tiene que ver con las técnicas utilizadas para el aprendizaje de la máquina. Se ha observado cómo el uso de la librería *libsvm*, frente a las funciones *svmtrain* y *svmclassify* que proporciona Matlab, aumenta el rendimiento general, haciendo el entrenamiento más eficiente ya que se ha apreciado que el tiempo de simulación se reduce de manera considerable en la segunda aproximación al problema, pese a tener una cantidad de datos mucho mayor. En resumen, el software *libsvm* se hace muy recomendable para tareas de clasificación basadas en SVM.

En referencia a la creación del conjunto de datos, basado en archivos de audio, cabe destacar el acierto al decidir un cambio de estrategia en el segundo conjunto de experimentos, frente al primero. El hecho de utilizar las canciones completas en lugar de segmentos de canciones, ha provocado unos resultados estadísticamente más razonables y cercanos a la realidad. El aumento en número y heterogeneidad de las observaciones también ha resultado importante de cara a la consecución de los objetivos. Se ha demostrado también la importancia de elegir un conjunto estadísticamente significativo, en el cual las observaciones tengan muy diversa naturaleza emotiva, para conseguir un entrenamiento más completo, y así la máquina pueda aprender de datos más dispares.

Tras observar los resultados del segundo conjunto de experimentos, lo primero que llama la atención es el aumento del error del clasificador. Como se ha explicado en el Capítulo 4, todo indica a que es debido a la disparidad de las observaciones. En un conjunto tan amplio de canciones, se echa de menos alguna clase más en la que etiquetar las emociones, ya que puede que algunas de las canciones no posean características emotivas propias de alegría o tristeza. Este factor también puede incidir directamente en la disminución de la tasa de acierto del clasificador, si durante el etiquetado hubo canciones en las que no estaba clara la emotividad. Una vez eliminadas las características ruidosas y redundantes que suman, en promedio, 27 características, las cuales entorpecen la clasificación, se alcanza un mínimo de error promedio de 38.6%. Efectivamente, la tasa de acierto no resulta ser tan buena como se esperaba, pero se deben tener en cuenta los factores explicados anteriormente.

Contrarrestando este hecho, los rankings de relevancia de características han resultado muy estables en la segunda fase de experimentos. Varias simulaciones independientes con un número elevado de iteraciones en cada una (100, 500 y 1.000 iteraciones) han ofrecido un orden de relevancia casi idéntico, pudiendo asegurarse que las características presentes en los primeros puestos son, sin lugar a dudas, decisivas en el carácter emocional de las canciones.

5.2. Características más relevantes en la emotividad de una canción

En primer lugar, la posición de los picos de fluctuación parece ser el aspecto más relevante en la emotividad de una canción. La fluctuación se definió como la “periodicidad rítmica a lo largo de los canales auditivos”. Resulta lógico pensar que una canción alegre pueda presentar un

mayor número de picos en el ritmo, mientras que una canción triste tenderá a ser más lineal en este sentido.

La segunda característica más importante para el clasificador es la irregularidad del espectro de una canción, o el grado de variación de máximos sucesivos en la representación frecuencial de la señal de audio. De alguna manera, esta característica es la análoga a la primera, pero en su forma frecuencial. Si la variación de frecuencia durante la canción es grande, se manifiesta en un ritmo cambiante, más propio de canciones alegres. En canciones tristes, por el contrario, los cambios de frecuencia son menos numerosos.

El tercer puesto en relevancia para el clasificador lo ocupan los coeficientes cepstrales en las frecuencias Mel, o más comúnmente conocidos como MFCCs (Mel Frecuency Cepstral Coefficients). Se trata de coeficientes basados en la percepción auditiva humana, normalmente relacionados con el habla, utilizados ampliamente en reconocimiento de audio automático para la identificación de estados de emoción en interlocutores y la búsqueda de contenido relevante.

La fuerza de las notas predominantes en la canción, es, según el clasificador, la cuarta característica que toma más importancia en la decisión sobre la emotividad. Esto puede explicarse aplicando la teoría musical, acercando la idea de 'notas predominantes en la canción' a la definición de acorde musical. Un acorde es una combinación armónica de tres o más notas, tocadas simultáneamente por uno o varios instrumentos, incluida la voz. Entre los tipos básicos de acordes, se encuentran los acordes mayores y acordes menores. Los acordes mayores siempre se asocian a alegría, mientras que los acordes menores sugieren, en general, melancolía o tristeza. Por tanto, un número predominante de acordes mayores o menores puede marcar la emotividad de una pieza musical.

La sexta característica a destacar es el flujo espectral, que está relacionada con la distancia entre eventos importantes en el espectro. En concreto, las fluctuaciones en el rango frecuencial de 1600Hz a 6400Hz corresponden a la percepción de la actividad de la música. De manera que, parece lógico pensar que canciones con mayor actividad en estas bandas sean buenas candidatas a canciones alegres, mientras que las canciones con emotividad triste tenderán a mostrar menos fluctuaciones en las frecuencias mencionadas.

Capítulo 6. Trabajos futuros

Para finalizar este proyecto, se ofrecerán tres líneas futuras de trabajo que pueden servir de continuación o ampliación del trabajo descrito en esta memoria, mejorando alguno de los aspectos del planteamiento, desarrollo o resultados.

6.1. Aumento del número de emociones a clasificar

Como se ha descrito en el capítulo anterior al hablar del problema del etiquetado en un conjunto de canciones de número elevado, sería altamente recomendable estudiar cómo afectaría a los resultados la inclusión de un número mayor de emociones en las que clasificar las observaciones. Las emociones básicas inducidas por la música se podrían ampliar a cuatro, añadiendo, a las estudiadas en este proyecto, las emociones de miedo (o agresividad) y tranquilidad. Son múltiples los escenarios de emotividad que se pueden plantear. Por ejemplo, un estudio encabezado por Thomas Fritz, del instituto Max Planck de Alemania, enumera las tres emociones básicas de la música, que cualquier individuo puede identificar, que son felicidad, tristeza y miedo. El estudio asegura que “estas tres reacciones emocionales forman parte de un lenguaje universal, de una manera similar a las expresiones faciales humanas” [15].

6.2. Etiquetado de las observaciones mediante encuesta

En el Capítulo 4 se ha explicado el método de etiquetado, así como las razones para trabajar con la simplificación de un etiquetado realizado por el propio alumno. En las conclusiones finales, se ha expuesto la posibilidad de que este hecho pueda incidir en los resultados de la clasificación. Se propone, por lo tanto, un método más representativo para el etiquetado. Una posible solución pasaría por utilizar una aplicación, ya sea mediante un interfaz web o aplicación móvil, para ofrecer la reproducción de las canciones en la que serían los propios “usuarios” los que etiquetan la emotividad de las mismas. Los resultados se ordenarían y se realizaría el etiquetado por mayoría de votos de emotividad en cada una de las canciones.

6.3. Emotividad como variable gradual

En este trabajo se ha mostrado una solución de clasificación en la que la salida de la decisión es de tipo duro, es decir, que sólo admite los valores -1 o 1, en función de si la variable ha tomado un valor por debajo o por encima de cero, para clasificarla en una de las dos clases disponibles. Sin embargo, el carácter alegre y triste de una canción es una variable gradual, no binaria. Una ampliación del presente clasificador pasaría por mostrar la salida blanda, es decir, el número real que devuelve la SVM y sobre el que se aplica el umbral de decisión en cero para convertirla en binaria. De esta forma, se podría estudiar, para cada canción, en qué punto se encuentra en una escala de emotividad desde el carácter triste al alegre.

Referencias

[1] CCRMA MIR workshop 2011 https://ccrma.stanford.edu/wiki/MIR_workshop_2011

[2] EarSketch. Teaching Computers to listen: 1. Music Information Retrieval http://earsketch.gatech.edu/learning/teaching-computers-to-listen-1-music-information-retrieval

[3] Descripción y Recuperación de Información Musical y Sonora (DRIMS) http://mtg.upf.edu/projects/drims

[4] ISMIR. The International Society for Music Information Retrieval http://www.ismir.net/

[5] MIREX. The Music Information Retrieval Evaluation eXchange http://www.music-ir.org/mirex/wiki/MIREX_HOME

[6] MIRToolbox https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

[7] Brain Tunning: Tunning the Brain For Music http://www.braintuning.fi/

[8] Christopher M. Bishop. Capítulo 4: Pattern Recognition and Machine Learning.

[9] Gustavo A. Betancourt. Las Máquinas de Soporte Vectorial (SVMs). Scientia et Technica Año XI, No 27, Abril 2005.

[10] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[11] Cerdá, Salvador; Segura, Jaume; Barba, Arturo; Cibrián, Rosa; Montell, Radha, Giménez, Alicia. Análisis mediante escalado multidimensional de las características musicales y los parámetros objetivos en salas virtuales. 44º Congreso Español de Acústica. Valladolid, 2013.

[12] Olivier Lartillot: “Mirtempo: tempo estimation through advanced frame-by-frame peaks tracking”. Finnish Centre of Excellence in Interdisciplinary Music Research. University of Jyväskylaä.

[13] Panda, Renato; Paiva, Rui Pedro: “Using Support Vector Machines for Automatic Mood Tracking in Audio Music. University of Coimbra, Portugal. Mayo 13, 2011.

[14] FFmpeg http://www.ffmpeg.org/

[15] Thomas Fritz, Sebastian Jentschke, Nathalie Gosselin, Daniela Sammler, Isabelle Peretz, Robert Turner, Angela D. Friederici and Stefan Koelsch: “Universal Recognition of Three Basic Emotions in Music”. Max Planck Institute for Human Cognitive and Brain Sciences, Leipzig, Germany. Abril 14, 2009.

Anexos: Canciones

Anexo A: Canciones seleccionadas para el conjunto 1

1. Bohemian Like You – The Dandy Warhols
2. I'm Gonna Be – Proclaimers
3. I Want You Back – The Jackson Five
4. Jamming – Bob Marley & The Wailers
5. Black – Pearl Jam
6. Creep – Radiohead
7. Hurt – Johnny Cash
8. Yesterday – The Beatles
9. Brown Eyed Girl – Van Morrison
10. Hope There's Someone – Andy & The Johnsons
11. I'll Find A Way – Rachel Yamagata
12. Sleeping In My Car – Roxette
13. The Way You Make Me Feel – Michael Jackson
14. Wonderwall - Oasis

Anexo B: Canciones seleccionadas para el conjunto 2

1. Rock And Roll – Led Zeppelin
2. Heartbreaker – Led Zeppelin
3. Bombtrack – Rage Against The Machine
4. Break on through – The Doors
5. Help – The Beatles
6. Old Black – Earth
7. Open Mind – Blackfield
8. People of the sun – Rage Against The Machine
9. Set guitars to kill – And So I Watch You From Afar
10. Shroud of false – Anathema
11. Solitary One – Paradise Lost
12. Mellon Collie And The Infinite Sadness – The Smashing Pumpkins
13. Black Betty – Ram Jam
14. Ol'55 – Tom Waits
15. Stiff Upper Lip – AC/DC
16. By The Way – Red Hot Chili Peppers
17. S Is For Salamander – And So I Watch You From Afar
18. Angels – Black Mountain

19. Boston – Augustana
20. Blackfield – Blackfield
21. Bulls On Parade – Rage Against The Machine
22. The Scientist – Coldplay
23. Daddy Cool – Boney M.
24. Erase / Rewind – The Cardigans
25. Fragile Dreams – Anathema
26. La rubia perfecta – CanteCa de MaCao
27. Reptilia – The Strokes
28. Starlight – Muse
29. Tonight, Tonight – The Smashing Pumpkins
30. Stuck In A Moment You Can't Get Out Of – U2
31. Woman – Wolfmother
32. Act of love – Neil Young
33. Good Day Today – David Lynch
34. Misirlou – Dick Dale & His Del- Tones
35. Pictures of you – The Cure
36. Song 2 – Blur
37. Strays – Jane's Addiction
38. Ashtray Heart – Placebo
39. I Bet You Look Good On The Dancefloor – Arctic Monkeys
40. Life is a bullet – Papa Roach
41. Alive – Pearl Jam
42. The Path – HIM
43. The Boss – James Brown
44. Kids will be skeletons – Mogwai
45. La valse d'Amélie – Yann Tiersen
46. Primavera del 87 – La Fuga
47. Nude – Radiohead
48. Dark Night – The Blasters
49. Dollar Bill – Screaming Trees
50. Low – Blueneck
51. Scar Tissue – Red Hot Chili Peppers
52. Minerva – Deftones
53. Nube de pegatina – Los Delinquentes
54. Come as you are – Nirvana
55. Rain – Guano Apes
56. When one eight becomes two zeros – Glassjaw
57. 3 libras – A perfect Circle
58. Honesty in Death – Paradise Lost
59. Killing all the flies – Mogwai
60. The Hellcat spangled shalalala – Arctric Monkeys
61. Why go – Pearl Jam
62. Wish you were here – Pink Floyd
63. Whithout You – David Bowie
64. Wucan – Black Mountain
65. Comptine D'un Autre Été: L'après Midi – Yann Tiersen
66. The holy ghost – Crosses

67. Exit Music (For a film) – Radiohead
68. How to disappear completely – Radiohead
69. Little green bag – George Baker
70. Princeton review – Team Sleep
71. Stop Whispering – Radiohead
72. Steal My Soul – Airbag
73. Stepping Stone – Duffy
74. My Immortal – Evanescence
75. La primavera trompetera – Los Delinquentes
76. The River – Bruce Springsteen
77. Walking Thru Barbed Wire – Papa Roach
78. El Viento – Empty Space Orchestra
79. Black – Pearl Jam
80. Love Hurts – Incubus
81. Mr. Zebra – Tori Amos
82. New Dawn Fades – Joy Division
83. Rasputin – Boney M.
84. Re-connect – Anathema
85. Walking Through Walls – Vessels
86. A Means To An End – Joy Division
87. Lejos De Tu Cama – Havalina
88. 11am – Incubus
89. Stylo – Gorillaz
90. Like A Mirror – Morphine
91. Let Spirits Ride – Black Mountain
92. Decompression Period – Papa Roach
93. Like a Stone – Audioslave
94. Abrázame – La Fuga
95. Contigo – CanteCa de MaCao
96. Electric Worry – Clutch
97. Know Your Enemy – Rage Against The Machine
98. Lowlands – The Flying Eyes
99. Ma Baker – Boney M.
100. Six Different Ways – The Cure
101. Back In Black – AC/DC
102. I Know – David Lynch
103. Karma Police – Radiohead
104. Ley De la Gravedad – Havalina
105. Love, Hate, Love – Alice In Chains
106. Losing My Religion – R.E.M.
107. One Last Breath – Creed
108. Sueños – La Fuga
109. La Pared – Havalina
110. Lilitu – Blueneck
111. Little Sister – Queens Of The Stone Age
112. Vida maquinal - Havalina
113. What Happened Yesterday – Neil Young
114. You Shook Me All Night Long – AC/DC

115. Mercy – Duffy
116. Cloudy Now – BlackField
117. The Reason – Hoobastank
118. Judgement – Anathema
119. My Favourite Game – The Cardigans
120. Regret – Anathema
121. The End – The Doors
122. Yuki – Vessels
123. Coda Maestoso In F(flat) Minor – Earth
124. Destroy The Map – 36 Crazyfists
125. Revelations – Blueneck
126. Porcelain – Red Hot Chili Peppers
127. Godsmack – Alice In Chains
128. Don't Waste Time Doing Things You Hate – And So I Watch You From Afar
129. La Chispa Adecuada – Héroes Del Silencio
130. Manhattan Project – Truckfighters
131. One Second – Paradise Lost
132. Run – Snow Patrol
133. Televators – The Mars Volta
134. Ice Cream Man – Tom Waits
135. Habit – Ben Kenney
136. Cuando Todos Duermen – Havalina
137. Emotional Winter – Anathema
138. Videotape – Radiohead
139. Lonely Day – System Of A Down
140. I Can't – Radiohead
141. Stuck In The Middle With You – Stealers Wheel
142. Slow Numbers – Morphine
143. You Make Me Smile – Blue October
144. Are You In – Incubus
145. When The Sun Goes Down – Arctic Monkeys
146. El aire de la calle – Los Delinquentes
147. Get Back – The Beatles
148. Yesterday – The Beatles
149. Dirty Frank – Pearl Jam
150. Lucky Man – The verve
151. Felicidad – Boney M.
152. Hurt – Johnny Cash
153. Vertigogo – Combustible Edison
154. Everybody hurts – R.E.M.
155. We Are The Champions – Queen
156. Why Did You Do It – Stretch
157. Requiem For A Dream OST – Marion barfs
158. La Maison – Yann Tiersen
159. Close To Me – The Cure
160. EULA – Baroness
161. I Shot The Sheriff – Bob Marley
162. Jammin' – Bob Marley

163. Sun Is Shining – Bob Marley
164. Buffalo Soldier – Bob Marley
165. Blue – A Perfect Circle
166. Rowboat – Coal Chamber
167. Honestly OK – Dido
168. My Life – Dido
169. High Voltage – Eagles Of Death Metal
170. Senderos de este infierno – El Ultimo Ke Zierre
171. All My Life – Foo Fighters
172. Tired Of You – Foo Fighters
173. November Rain – Guns N' Roses
174. American Idiot – Green Day
175. Good Riddance (time Of Your Life) – Green Day
176. Prosthetic Head – Green Day
177. This is a mans World – James Brown
178. Aint no mess – James Brown
179. In The End – Linkin Park
180. Bodyrock – MOBY
181. Honey – MOBY
182. Machete – MOBY
183. Natural Blues – MOBY
184. Ace Of Spades – Motorhead
185. Stones From The Sky – Neurosis
186. Breed – Nirvana
187. Drain You – Nirvana
188. Go Let It Out – Oasis
189. Broken Home – Papa Roach
190. Last Resort – Papa Roach
191. English Summer Rain – Placebo
192. This picture – Placebo
193. The bitter end - Placebo
194. Special Needs – Placebo
195. Sunburn – Muse
196. Muscle Museum – Muse
197. Roxanne – Sting
198. Every Breath You Take - Sting
199. Bother – Stone sour
200. Friday I'm in love – The Cure